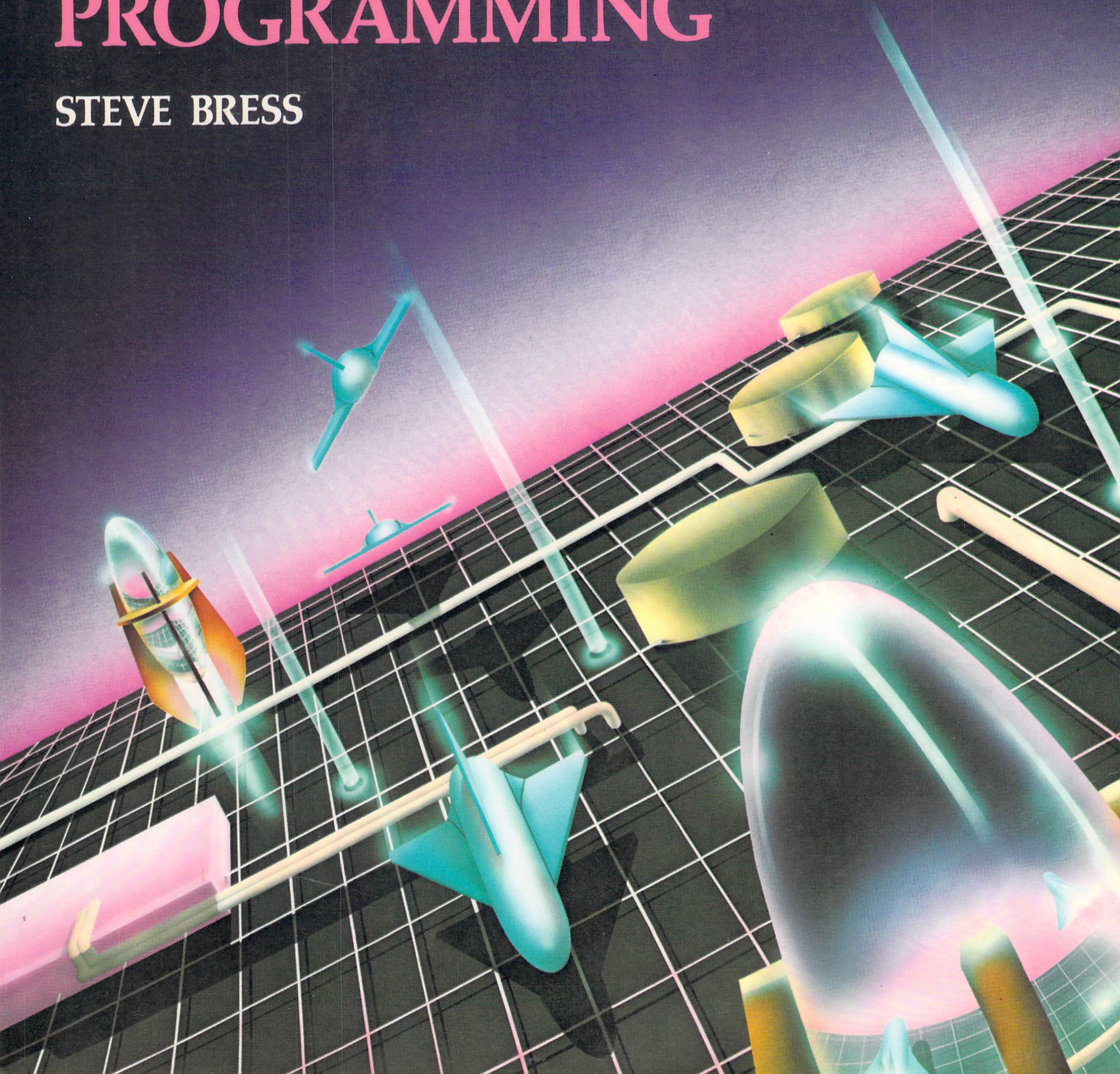
1919

COMMODORE 64 ASSEMBLY LANGUAGE ARCADE GAME PROGRAMMING

STEVE BRESS



COMMODORE 64
ASSEMBLY LANGUAGE
ARCADE GAME
PROGRAMMING

This book is dedicated to my parents.

COMMODORE 64 ASSEMBLY LANGUAGE ARCADE GAME PROGRAMMING

STEVE BRESS

TAB TAB BOOKS Inc.
Blue Ridge Summit, PA 17214

Game design by Dan and Steve Bress.
Illustrations by Elisa Frances Mosely.
Revenge of the Phoenix artwork by Denise McDonald.

PAC-Man is a trademark of Bally, Midway Mfg. Co.
DONKEY KONG is a trademark of Nintendo.
CENTIPEDE is a trademark of ATARI, Inc.

FIRST EDITION

FIRST PRINTING

Printed in the United States of America

Library of Congress Cataloging in Publication Data

Bress, Steve.
Commodore 64 assembly language arcade game programming.

On t.p. the registered trademark symbol "TM" is superscript following "64" in the title.

Includes index.
1. Commodore 64 (Computer)—Programming. 2. Assembler language (Computer program language) I. Title.
QA76.8.C64B73 1985 001.64'2 85-2803
ISBN 0-8306-0919-9
ISBN 0-8306-1919-4 (pbk.)

Contents

Introduction viii

1—You and Your TV 1

How a Computer Displays a Picture 1
What Is Animation 2

2—A Language for Games 3

Assembly Language 4
Using an Assembler 4
What an Assembler Can Do for You 5
How to Choose an Assembler 6

3—Underlying Concepts 9

Bits and Bytes 9
The Hardware 10
6510 Architecture 11

4—The 6510 Assembly Language 13

Instruction Types 13
Addressing Modes 14
Immediate Mode Addressing—Zero Page Addressing—Zero Page Indexed Addressing—Absolute Addressing—Absolute Indexed Addressing—Indirect Addressing with Indexes—Implied Addressing—Relative Addressing—Indirect Addressing

5—Organizing Your Program 17

6—Working with Interrupts 19

7—Technical Information 21

Commodore 64 Address Space 21
Memory Control and Mapping 22
Graphics Memory Locations 23
Standard Text Mode 24
Color Memory 27
Custom Character Sets 28
Multicolor Mode 28
Extended Background Color Mode 28
Bit Mapping 29
Multicolor Bitmapped Mode
Sprites 30
Sprite Pointers—Sprite Controls
Collision Detection 32
Blanking the Screen 33
The Raster Register 33
Video Interrupts 33
Scrolling 33
Joysticks 34

8—Sound Effects 36

Filtering 36
The Sound Generator Demo 41
The Sound Editor 41

9—Creating Graphics 47

Hand Coding Graphics 47
Using a Graphics Tablet 52
Using a Koalapad
Using the Sprite Maker 55
Using the Screen Maker Utility 56

10—Some Arcade Games 57

Pac-Man 57
Donkey-Kong 58
Centipede 58
Revenge of the Phoenix 59
Game Play—Scoring

11—Elements of Game Design 61

Visual Impact 62
Sound Effects 62
Difficulty Levels 62
Scoring 63

12—How Boghop Works 64

The Start of the Program 65
The Macro Library—RAM Definitions—Musical Definitions—The Data Section—The Point Plotting Routine—The Equates Statements—Defining the System—Initializing RAM
The Main Program Loop 70
Scores—Moving the Bad Guys—Incrementing the Level of Play—Seeing if the Bad Guy Is Hit—Seeing if the Player Is Hit—Launching the Shots—Moving the Bad Guy's Shots—Moving the Player's Shots—Checking the Shot Positions
Animating the Sprites 73

Displaying the Number of Lives—Resetting the Game—Maintaining the Timing
The Interrupt Routines 74
The Player Controls—Clear and Pull—INT1—Moving Mountains
The Movement Subroutines 76
An End Note 76

Appendix A—The Instruction Set 77

Appendix B—The Macro Library 99

Appendix C—The Program Listings 119

Glossary 254

Index 257

Introduction

This book, written for those who want a greater understanding of their computer's operation, is recommended for programmers who are familiar with the essential concepts of machine language and experienced in programming home computers in BASIC. It is also for machine language programmers. Many of the examples given in the book are machine language subroutines that can be used by machine language programmers or called from BASIC to increase the speed of BASIC programs.

This book is divided into three sections:

- Technical information
- Game and graphics design
- A description of the code for a game program

After you have mastered some of the programming techniques, you will be shown how to take a game concept and turn it into a video game. Many of the routines presented in this book were created expressly for use in video game programming. However, in many cases techniques used in programming video games carry over into all other fields of programming. By experimenting with the new concepts and techniques as you read, you will find that you are increasing your understanding of how to fully utilize your Commodore 64.

Chapter 1
You and Your TV

Virtually every home in the country has at least one television set, and televisions, like many other domestic appliances, tend to be taken for granted. Most people do not have any idea of how they work. Since you are interested in having the television display your ideas and visions in the form of a video game, you should have some idea of how an image is generated on the television screen to best utilize the capabilities of your computer.

HOW A COMPUTER DISPLAYS A PICTURE

The face of a video display is coated with special phosphors. An electron beam strikes the face of the tube causing the phosphors to glow. On a black and white monitor, this process produces a dot. This glowing dot is called a *pixel.* A pixel is the smallest area that a computer can control on the screen. A color monitor uses three electron beams and three different colored phosphors to create one pixel.

The electron beam scans from left to right across the face of the monitor. By controlling the intensity of the electron beam during its scan, different points on the screen receive different intensities.

When the beam gets to the right edge of the screen, a horizontal sync pulse causes the beam to cut its intensity and return to the left side of the screen on the next line down. This process is repeated 262-1/2 times to form one display screen. At the end of the screen, a vertical sync pulse is initiated. During the vertical sync pulse, the beam returns to the upper left side of the screen and the whole process is ready to start again. On the Commodore 64, the computer's hardware automatically inserts both the horizontal and vertical sync pulses, so the programmer need not worry about generating them.

When a Commodore 64 is driving the monitor, 200 scan lines are used to display text and graphics with the other 62 lines being used for the border. The display process is repeated 60 times per second, providing a flicker free display.

There is a time correlation between the speed of the microprocessor and the position of the beam

on the face of the screen. Some special effects can be created by using this fact and changing the display parameters on the fly. In the time it takes for the microprocessor to go through one machine cycle, the beam travels approximately 6 pixels on the display screen.

WHAT IS ANIMATION

As you were reading the description of the generation of a TV *frame*, you may have noticed that the only thing that a monitor can display is a series of still frames. Thus the question of how you can get animation out of still pictures arises.

There is a characteristic of the human eye called the *flicker fusion frequency*, which allows us to view TV shows and movies without seeing that they are made up of still frames. This frequency is 24 frames a second. Any time a series of pictures is shown at a rate faster than 24 *hertz*, the eye can no longer distinguish the individual pictures. If the computer makes small changes in its display faster than 24 times per second, these changes will give the appearance of being continuous.

It is important to understand that the programs you will be writing create a series of still frames, not continuous motion. Because the computer updates the screen 60 times per second, this is the fastest that any changes can occur on the screen. If an object is in motion at the speed of one pixel per screen change, it takes about 5.3 seconds for the object to get from one side of the screen to the other. If the object needs to go faster, it has to move more than one pixel per screen update. On the other hand, if the object is to go slower, it has to stay still during some screen updates.

Because the video monitor provides a known minimum update time of 1/60 second, this tends to become the time period by which most aspects of the game are measured. This period of time (1/60 second) will be called a *screen*.

From what you have read, you might assume that completely different displays could alternate 30 times per second, and the eye would fuse them. This is true; *multiplexing* is the term used to refer to this technique. Some care must be taken when you attempt to use this technique. The differences between the two screens should not be excessive, and best results are achieved using small fast moving objects.

Chapter 2
A Language for Games

The first question that arises when you are starting work on a computer project is, "What language should I program in?" Your first inclination might be to use BASIC. However, if your program is using any type of continuous motion, BASIC would most likely be too slow to be useful. If you try to write anything more than the most simplistic game in BASIC, you will find the motion of the objects slow and erratic. There are limits to how quickly sounds and colors can be changed.

All of these problems arise because the BASIC interpreter controls the computer, not the program. Every line in a BASIC program must be analyzed, decoded, and translated into a series of machine language instructions EVERY time the line is encountered. BASIC is also a very general language in that similar lines may have different functions. By the time a line is analyzed, the resulting section of machine code is not efficient. For all of these reasons, a BASIC program will never run particularly fast.

To achieve smooth animation, the program must be able to update the screen at least 30 times per second. Because BASIC does not have an easy way to talk to the hardware registers, (PEEK and POKE commands must be used), calculating new positions and updating the registers can easily take longer than 1/30 second, causing erratic motion. This same problem occurs when the sound registers must be updated. There may not be enough time to make a consistent sound.

Machine language, on the other hand, is the most efficient form that a program may take. Each instruction in the program is executed exactly as it is entered. There is no interpretation done on the code, so it runs at the highest possible speed. You also know the status of the entire computer at every step—which rarely happens in a BASIC program.

The major drawback to programming in machine language is that it is a collection of hexadecimal codes. This is fine for the computer, because the binary data that these codes represent can be immediately executed, but it is cumbersome for the programmer. Most programmers do not en-

joy memorizing all of the hex codes, and even fewer enjoy reading such a program when it is finished.

ASSEMBLY LANGUAGE

The alternative to these two extremes is to program in *assembly language*. Assembly language is a language that uses an assembler that translates the *mneumonics* (memory aids) for the CPU's instruction set into the binary data that the processor can execute. This translation takes place once before the program is run, so the final program will be machine code. You end up with all of the speed advantages of a machine language program with none of the headaches. A properly written assembly language program is just as efficient as its machine language counterpart, so there is no reason to program in machine language if you can gain access to an assembler.

With an assembly language program, the programmer normally has more than enough time to do all of the calculations and updates to keep the animation constant and smooth. The program can also make use of all of the hardware features of the computer to create effects that are not possible through BASIC. When you are using assembly language, the computer is completely under software control, making it possible to determine what the computer is doing at all times. An assembly language program executes faster than any other type of program, which makes assembly language the language of choice for programming games.

Learning to program in assembly language is not as traumatic an experience as most programmers would have you believe. Unlike BASIC, the machine will always be doing exactly what you tell it to do. The main difference between BASIC and assembly language is that in assembly language you must keep track of where the data is in memory and where it must go. BASIC keeps track of variables for you, but it won't tell you where they are. In assembly language, you can define various memory locations so that they have some meaning to you.

For instance, you could define $20 to be the player horizontal position. ($ indicates a hexadecimal number; $20 is equal to 32 in the decimal system. See Chapter 3 for more information.) Whenever you need to find out what the horizontal position is or need to modify it, you would look into location $20. You can create your own variables in BASIC using the same technique. You would POKE the value into $20 and PEEK it back out whenever you needed it. This is a particularly useful technique when you mix BASIC and assembly language, because each program will know where to find the data from the other program.

USING AN ASSEMBLER

The rest of this book assumes that you will be programming in assembly language, and most of the examples are written in assembly language. If you are a BASIC programmer, you may be able to use some of these routines to speed up parts of your programs.

Each assembler has its own set of *pseudo opcodes*—the instructions that tell the assembler to do something other than create code. In order to ensure that you will be able to use these routines, the following is a listing of the pseudo-opcodes used by Commodore's Macro Assembler Development System, which is the assembler used in this book. By modifying these instructions to match those of your assembler, you should be able to run any of the examples in this book. Normally, you can type in one of these instructions anywhere that it is legal to type in one of the CPU's opcodes.

.BYTE	Reserves one or more bytes of data starting at the current location counter value
.WORD	Reserves 16 bit data in a LOW byte-HIGH byte format
.DBYTE	Reserves 16 bit data in a HIGH byte-LOW byte format
*	Program location counter
.LIB	Insert another disk file following this command
.END	End of file marker
=	Assigns a value to a symbol

<	Specifies the low order 8 bits of a 16 bit value
>	Specifies the high order 8 bits of a 16 bit value
.MAC	Starts a macro
.MND	Ends a macro

All of the above may be preceded by a label.

?	Precedes a number that specifies which parameter to pass to the macro. It can also be used as a label.

These are all of the commands that may be different from those in your assembler. With this list, you should be able to read all of the program listings and modify them to work with your assembler. Many assemblers come with a program that will translate files with this syntax into their own syntax. If your assembler has one of these programs, you will not have to make many changes by hand to assemble the listings on the distribution disk.

WHAT AN ASSEMBLER CAN DO FOR YOU

An assembler relieves you from memorizing the actual machine codes for each of the instructions. It also calculates the distance from one instruction to another for those times when a branch must be taken from the main program. This is not a particularly big deal for a short program (under 50 lines), but as a program grows larger and more complex, the task of repeatedly doing these calculations by hand becomes unreasonable. (If you are the type of person who finds great enjoyment in hand coding machine language programs, let me apologize here for suggesting there is a better way. The rest of us will let the computer do the tedious jobs).

The most useful feature of an assembler is its ability to let you assign names. A name can be given to a memory location, hardware registers, or a location within the program. Once names have been assigned, it is no longer necessary to remember long lists of confusing addresses. You only need to remember the names you have assigned to the addresses. Since you will normally assign names that carry some meaning (at least to you) to the memory locations and program segments, the program becomes infinitely more readable than if the addresses themselves had been used.

But wait, did that last sentence use "readable" when referring to an assembly language program? Yes it did. Assembly language programs become illegible to others because it is rare to see a full listing of the program with all of its definitions; and, often a disassembly of a program is called the original program. (A disassembly has no legitimate names, just addresses.)

All good assemblers also have the ability to use *macro-instructions (macros).* A macro is a shorthand notation that represents a series of assembly language commands. For example, a macro that increments a two byte value by a one byte value could be coded as follows:

```
.MAC  DBINC  ;REGISTER NAME, DATA
LDA   ?1     ;LOAD THE LOWER BYTE
CLC          ;CLEAR THE CARRY BIT
ADC   #?2    ;ADD WITH CARRY THE DATA
STA   ?1     ;STORE THE LOWER BYTE
LDA   ?1+1   ;LOAD THE UPPER BYTE
ADC   #$00   ;ADD THE CARRY BIT
STA   ?1+1   ;STORE THE UPPER BYTE
.MND         ;END OF MACRO
```

This macro is used to increment any two consecutive bytes, such as a score. If the score needed to be incremented by $20, you would type:

```
DBINC SCORE,$20
```

which would be expanded by the assembler to read:

```
LDA  SCORE
CLC
ADC  #$20
STA  SCORE
LDA  SCORE + 1
ADC  #$00
STA  SCORE +1
```

This is certainly easier than typing the same

series of instructions every time that you need to increment a two byte value. A program that uses macros will be easier to read as you work on it than one written using individual instructions. Once you have written and debugged a macro, it becomes a tool that can be quickly used whenever necessary. By building up a library of macros, you will be able to program quite difficult functions in a minimal amount of time.

There are many cases in which it is a better idea to use a subroutine instead of a macro. Each time a macro name is entered into a program, the assembler expands it into its individual instructions. This means that each time the macro is called, it is treated as if you have entered all of its instructions by hand, and uses the same amount of memory as if you had.

For a function that is repeatedly used and takes a large amount of code, it is better to use a subroutine. A subroutine is called using the JSR instruction and is only stored once in the program. If you write a routine to display text on the screen that many different parts of the program are going to be calling on often, it is best treated as a subroutine. You may build up a library of subroutines in the same manner in which you build up a library of macros.

Normally, in the course of working on a game you will run into a situation that requires you to write a specialized routine to perform a certain function. If you think that you will be able to use this function at a later time, you should incorporate it into either your subroutine or macro library. This way, you will quickly have the major routines that are common to most programs at your fingertips.

An assembler must also provide some means of defining data areas and data. Tables of data can be defined, given a name, and stored by the assembler. A good assembler allows you to define data in terms of mathematical expressions. It also allows data to be defined as one or two byte values. There is a further option on two byte values as to whether the high byte or the low byte will be stored first. For many programs, text must be stored for later printing. On some assemblers, you have the option for text to be stored with the high bit on or off. This can be useful for finding the end of a text string.

Finally, an assembler must allow you to enter assembly language commands. After all, that is the point of an assembler.

HOW TO CHOOSE AN ASSEMBLER

If you are to successfully create your own machine language video games, you must become familiar with your primary design aid, the assembler program. Next to your computer, a good assembler program is the most essential tool for the creation of a machine language program.

There are three parts to a good assembler package:

- A text editor
- The assembler
- A machine language monitor

The text editor is the part of the assembler that you will spend the most time with. It allows you to enter, modify, and update your program. Some assemblers allow you to use a word processor to enter your program. Whatever method you choose, make sure that you are comfortable with the editing commands that are available on your text editor.

It is a good idea to try out an editor before buying it. Some assemblers come with editors that are very limiting in what they allow you to do. Limitations in the editor take from your programming time, so it pays to shop around for a good one.

Once the program has been entered into the editor, it must be assembled before it can be used. Some assembler packages force you to load the assembler at this point, while others already have it loaded. An assembler that has all of the programs you need loaded simultaneously is called a *coresident assembler.*

A coresident assembler can save you quite a bit of time if you like to write a small section of code and immediately try to assemble it to check for errors. If you have a coresident package, remember always to save your source code before attempting to run your program. You can lose all of your latest work if your new program locks up the computer,

forcing you to turn it off. In fact, no matter what type of assembler you are using, you should save your source code often as you are writing it.

Another aspect to examine while you are choosing an assembler is the speed with which it can assemble your source code and generate the necessary files on disk. Unfortunately, you can't expect the assembler to work faster than the disk can move the data.

You also should be sure that any printouts generated by the assembler contain all of the information that you would like. The following are some of the items that differ between assemblers:

- Sorted symbol table with absolute addresses
- Macro expansion
- Data expansion
- Absolute addresses for all code
- Absolute addresses for RAM registers

Depending on how you approach debugging your program, these different functions will have different levels of importance to you.

A symbol table is generated by all assemblers at some point. It is a list of the names used in the program and the addresses that correspond to the names. A printout of the table can help you verify the assembler is working properly and gives you a quick guide to any location used by the program. For this reason it helps if the machine sorts the table alphabetically before printing. On the other hand, if the assembler only prints out the symbol names and its internal representation of the addresses (usually filled in later by the assembler), the printout is useless for most purposes.

If you are going to use macros in your program, it is useful to have an assembler that lets you specify whether or not it should expand the macro before it is printed. When a macro is expanded for print, the macro name is expanded for print, the macro name is printed followed by the code it generates with all of the substitutions shown. On the printout, all of the instructions of the expanded macro are generally preceded by a + symbol. Without a macro expansion on the printout, you must constantly refer back and forth between a listing of your macro library and the section of code where the macro was called. This can be quite time consuming and prone to error as you expand the macro by hand for debugging purposes. But once all of your macros have been debugged and you are familiar with them, you rarely need to see them expanded on your printouts.

Since most assemblers allow you to use expressions in data statements, you should be able to get a printout of the calculated data. With such a listing, you can verify that the assembler generated the expected data. Again, once you are familiar with the operation of your assembler and your data has been debugged you rarely need to see this part of the printout.

Beware of an assembler that won't tell you where it has put your code or data. Your assembler should have in its printout absolute addresses for every instruction, data statement, and hardware or RAM register that has been used. If your assembler does not provide this information, you will find your program extremely difficult to debug.

The output of most assemblers is an intermediate file that contains all of the information needed about your program. This file is usually one form of a *hex file*. Hex files store the information about the program in a hexadecimal format that can be easily transmitted or loaded into the computer. (Binary data is more difficult to transfer from one machine to another.) You will use a program called a *loader* to translate the hex file into a binary file and place the data in the proper place in the computer.

One potentially useful option on most loaders is the ability to relocate the loading address of a program. For example, if you write a program to be placed on a cartridge, it needs to run from a different place in memory than if it is to be stored in RAM. The ability to relocate a program allows you to test it in one location although it is intended to run at another.

The last piece of an assembly language development system is a monitor. This is a program that allows you to examine the computer's memory and change or move the contents. It also must allow you to load and save areas of memory from the disk

drive. A good monitor has a small disassembler that allows you to view memory as assembly language commands.

A word of caution: it is not a good idea to get a monitor in a cartridge. A monitor on a cartridge resides in a permanently fixed place in memory. If your program needs to use this area, you can't use the monitor. In fact, if the monitor must reside in only one predefined place in the computer, it can be useless. Either the monitor should be relocatable, or you should be given two or more different versions of the monitor. If you have a version of the monitor that resides in high RAM and another version that stays in low RAM, usually you will be able to use one of them.

Chapter 3
Underlying Concepts

At this point, some of the essentials that you need to know in order to understand the rest of this book will be presented. The terms and names that will be used will be defined. The hardware of the Commodore 64, and in particular, a programming model of the 6510 microprocessor will be discussed. Also, the hexadecimal numbering system (base 16), which is used throughout this book, will be described. This numbering system makes the most sense when dealing with computers. Since an understanding of the hexadecimal numbering system and its relationship to bits and bytes will make the hardware descriptions easier to understand, it will be presented first.

BITS AND BYTES

A *bit* is the smallest meaningful piece of information that can be stored. It can have only two values, 1 or 0. Other terms for these values are shown below:

1	0
ON	OFF
SET	CLEAR
HIGH	LOW

All of the signals inside of the computer can only be in one of these two possible states. So how does the computer perform so many functions if it only has two states to work with?

By grouping a collection of bits together, the group of bits together can have a value that is equal to 2 raised to the power of the number of bits in the group. If we grouped 4 bits together, the group could have 16 possible values according to the equation 2^N where N is the number of bits in the group:

```
2^N  N=4
2^4
2*2*2*2=16
```

A grouping of 4 bits is called a *nibble*. Each of the

four bits is given a value depending on its position in the group. Spreading the bits out horizontally, the bit on the right is called the *least significant bit* (*LSB*). The bit on the left is called the *most significant bit* (*MSB*). The location of each bit in the byte is given a value of 2^N, where N is the number of bits from the LSB that the location in question is. For instance the LSB is located on itself so its distance (N) would be 0. Therefore, the LSB can have a value of 0 or 1 depending on the state of its bit. The next bit on the left would have a value of 2^1.

The group of four bits could have sixteen values, 0 through 15, as you have seen. As discussed earlier, the value of the bit depends on its location in the group and its state. To compute the value of the group of bits, you simply add together the value of each location in the group whose corresponding bit is ON. By summing the total of all of the positions in a group, the maximum value of a group can be determined. In the case of a nibble, the maximum value would be 15.

If 8 bits were grouped together, the equation would be:

```
2^N  N=8
2^8
2*2*2*2*2*2*2*2=256
```

So a group of 8 bits can have 256 different values. This grouping is referred to as a *byte*. Since 0 is the first of the possible values of a byte, the range of values is from 0 to 255. Inside the Commodore 64, all of the data is represented and transferred as bytes. This is what is meant when the C-64 is referred to as an *8 bit machine*. A byte is the standard unit of storage in the Commodore 64.

It was mentioned earlier that hexadecimal would be the standard notation to be used in this book. This is because one hexadecimal (*hex*) digit can represent 16 values or one nibble if it is representing a group of 4 bits. Thus it requires 2 hex digits to represent any 8 bit value or any byte. The correlation between the 4 bits of a nibble, its decimal value, and its hex value is shown in Table 3-1.

Table 3-1. The Relationship Between the Binary, Hex, and Decimal Number Systems.

2^3	2^2	2^1	2^0	DEC	HEX
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

You may notice that until the tenth value, (the number 9) the same numbers are used in both hex and decimal numbering systems. In hex however, the next 6 numbers are represented by the first 6 letters in the alphabet.

From this point forward, all hex numbers will be preceded by a dollar sign ($) to differentiate a hex number from a decimal number. This is the standard notation used by virtually all assemblers that assemble code for the 6500 series of microprocessors. If there is no $ preceding a number, it is assumed to be a decimal number, unless it contains any of the letters A-F, in which case you can assume that a mistake has been made.

THE HARDWARE

Like all computers, the Commodore 64 is made of a number of complex integrated circuit chips. You can conceptualize the internal workings of the computer as being broken down into 5 different sections:

- Central processing unit
- Memory
- Video generation
- Sound
- Input and Output

In the Commodore 64, the central processing unit (*CPU*) is a 6510 microprocessor chip. It executes the same instruction set as a 6502 microprocessor as used in Apple and ATARI computers. It runs with a clock frequency of 1.0225 MHz. For all practical purposes, this can be considered to be a 1 MHz clock. The 6510 has an addressing range of 65536 bytes (64K).

There are two different types of memory in the Commodore 64. It has 64K of dynamic RAM, which can be banked into the address space of the other chips as necessary. There is also 20K of ROM in the system. In this ROM are the BASIC programming language and the operating system of the Commodore 64. The operating system is responsible for reading the keyboard, updating the real-time clock, and transferring data in and out of the system, among other things. Since the CPU can only address 64K of memory, all of the RAM cannot be accessed simultaneously with all of the ROM. To overcome this problem, the technique of *bank switching* is used. For instance, if you are not using BASIC, there is no need for the BASIC ROM to be accessible. In this case, it can be replaced with RAM. The CPU cannot tell the difference, so it can be "tricked" into addressing more then 64K of memory.

Video generation is a task that is taken care of by a 6567 *Video Interface chip* (*VIC II*). All of the various graphic modes of the Commodore 64 are generated by this chip. In the process of generating the video signal, the VIC-II chip refreshes the dynamic ram chips used in the system. The VIC-II chip also generates the system clock from the 8.18 MHz dot clock.

Sound is generated by a 6591 *Sound Interface Device chip* (*SID*). This chip can generate 3 independent voices each in a frequency range of 0 to 4 kHz. This corresponds to a range of about 9 octaves. Each voice has an independent volume envelope and a choice of waveforms. The SID chip can also provide a number of filtering options for use with its own signals or an externally supplied signal.

Input and output functions are handles primarily by a pair of 6526 *Complex Interface Adapter* chips. Serial communication functions as well as the parallel port are maintained by these chips. They also handle input from the joysticks and the real time clock. These chips each provide a pair of independent 16-bit timers.

If you understand how these four devices work, you can make the computer do anything it is capable of. Your program will be primarily concerned with the VIC-II chip and the SID chip. The CPU is the chip that the program is written for, and it is directed to modify the registers in the other chips at the appropriate time for the intended function. Writing almost any type of program eventually comes down to controlling just a few chips. Once you control the major chips the rest of the program should be easy.

6510 ARCHITECTURE

In order to program in assembly language, you must understand the internal functions of the microprocessor. Figure 3-1 is a block diagram of the 6510. The value of the *program counter* is output on the *address* lines of the microprocessor whenever a data access is to be performed on the systems memory. In the Commodore 64, all of the hardware registers appear to be memory locations to the microprocessor, so accesses to hardware registers and memory appear identical.

The *accumulator* is the most important register in the computer. Almost all of the data that passes through the system goes through the accumulator. Every arithmetic function, other than incrementing and decrementing, is performed in the accumulator. Data can be read into the accumulator from memory, modified, and stored back into memory.

The *X* and *Y registers* are very similar. They move data in a manner similar to the accumulator. They can also be used as an index to an array of data. It should be noted that while these two registers are similar, their functions are not identical. Some instructions require the use of the X register while others use the Y register.

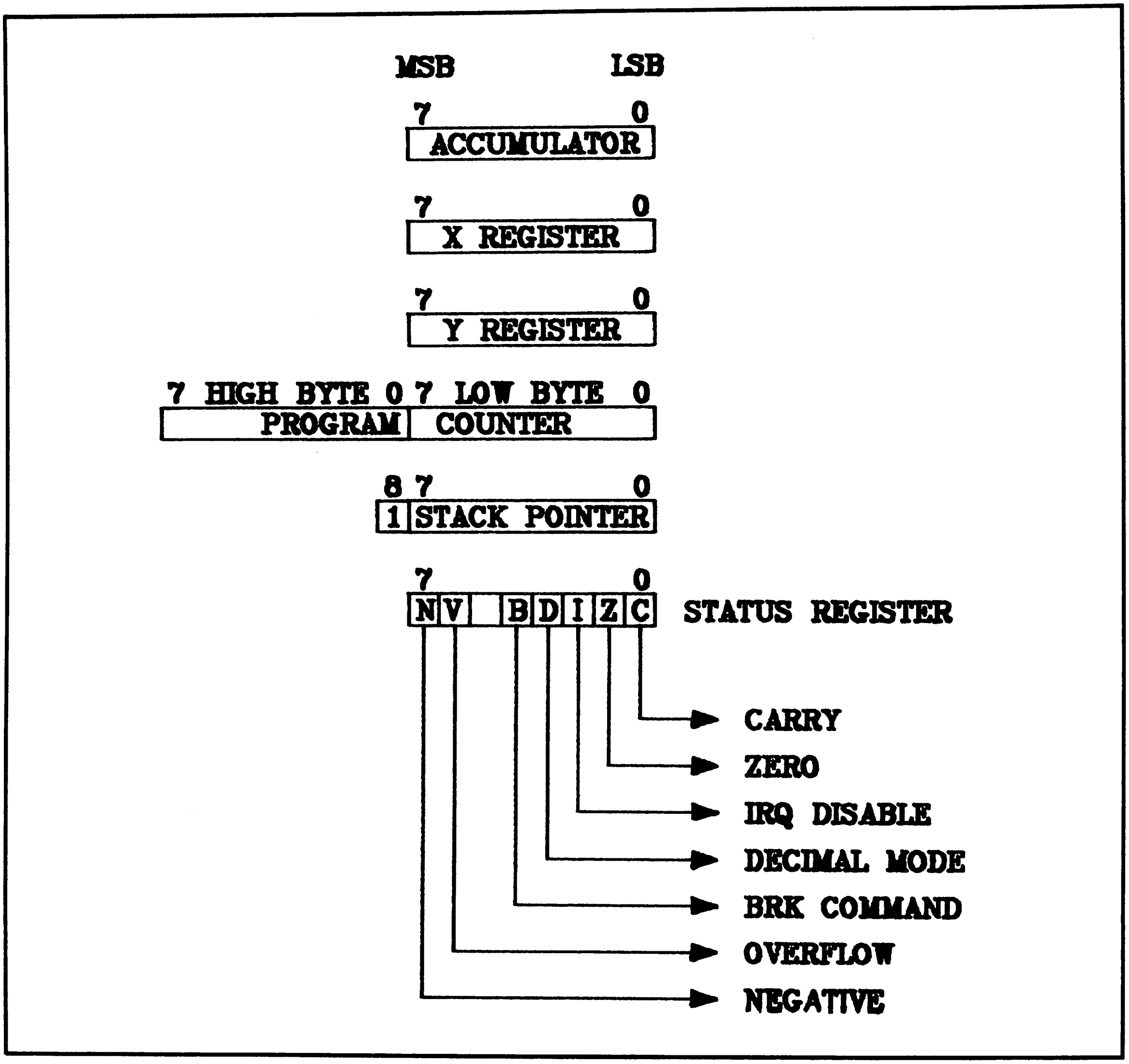


Fig. 3-1. Block diagram of the 6510 microprocessor.

Each of the bits in the *status* register correspond to one of the conditions in the microprocessor. Some of these bits can be changed under software control.

In the Commodore 64, the stack (a temporary data storage area between $0100 to $01FF) is controlled by the *stack pointer.* This register contains the address of the next empty space on the stack. At the start of a program the stack pointer is usually initialized to $FF, which corresponds to the top of the stack. The microprocessor defines the stack to start at $0100. The stack pointer is used as an index from the bottom of the stack.

With these concepts in mind, it is time to look at the language.

Chapter 4
The 6510 Assembly Language

In Appendix A are a series of charts that describe all of the instructions available in the 6510 as well as their addressing mode options. You may wish to refer to these charts as you are reading the following sections on instruction types and addressing modes. As you begin to program in assembly language, you will find yourself constantly referring to these charts.

All of the instruction lines used in assembly language take the following format:

LABEL OPCODE OPERAND ;comment

The *opcode* is the instruction that you want executed. The *operand* is the data, label, or memory address that will be operated on.

Comments are particularly useful in documenting your program and should be used often. A properly documented program is much easier to read and understand. A comment can either follow an instruction or be on a line by itself. A comment must be preceded by a semicolon.

Labels prevent assembly language from becoming unmanageable. A label must start in column 1 of the program line. A label can be assigned a value or can take on the value of the program counter during assembly. When used as the operand in a branch instruction, the assembler determines the length of a branch, which is much easier than calculating the branch distance by hand. Also, if a change is made in the program, the assembler can compensate for any changes in the branch. If the calculations are done by hand, they have to be redone every time there is a change.

INSTRUCTION TYPES

There are 4 classes of instructions in the 6510. These are:

- Data movement
- Arithmetic
- Testing
- Flow of control

Data movement instructions are instructions that cause a value to be loaded from memory, stored into memory, or transferred from one register to another. There are a number of options as to how the address of the byte to be loaded will be determined. In the load accumulator instruction, LDA, there are eight different addressing modes that can be used to determine which byte to load. The different addressing modes are explained in the following section.

Arithmetic instructions are used to modify data in some way. This class of instruction includes logical operations, such as the AND and ORA instructions. There are instructions that allow a byte to be rotated as well as addition and subtraction commands. As with the data movement instructions, most of the available addressing modes can be used by the arithmetic instructions.

Testing instructions allow a nondestructive test of data in the microprocessor. For instance, when a CMP instruction is used to check a value in the ACCUMULATOR, the data in the ACCUMULATOR will not be changed in any way. The bits in the STATUS register will be changed in the same way as if the data to be compared was subtracted from the ACCUMULATOR. These instructions are generally used to modify the STATUS register prior to executing a branch instruction.

Flow of control instructions are the branching and jump instructions. These are used to change the order in which different sections of code are executed. The branch instructions are all *conditional branching* instructions. That is, each instruction checks one of the bits in the status register and, depending on its value, will either branch to the instruction pointed to in the operand or execute the next instruction in line.

Jump and jump to subroutine instructions also fall into the flow of control category. These are known as *absolute commands* because they do not check any conditions before performing a jump.

ADDRESSING MODES

In the 6510 microprocessor, there are 11 types of addressing modes. They are:

Immediate	(Indirect,X)
Zero page	(Indirect),Y
Zero page,X	Implied
Absolute	Relative
Absolute,X	Indirect
Absolute,Y	

Many of the addressing modes can be used by the LDA, or load accumulator, instruction. This instruction causes a byte to be loaded into the accumulator.

Immediate Mode Addressing

In the immediate mode, the data that follows the # character will be loaded into the accumulator. Instead of entering data to be loaded, you could enter a label that has previously been equated to a value. For example, if you had defined the label BLACK to equal 0, the following statements would be identical:

```
LDA #$00
LDA #BLACK
```

In either case, a 0 is loaded into the accumulator. In all of the other modes, data is loaded from a memory location as indicated by the operand.

Zero Page Addressing

The *zero page of memory,* the memory locations in the range $00 to $FF, has a special meaning to the 6510. A location in this range can be accessed faster than anywhere else in memory. This results from the fact that the upper byte of the address will always be $00, so it need not be read from the operand during program execution. This also means that a program will be shorter because the upper byte of the address is not part of the code. Exclusive use of zero page memory can cut the program size and execution time by a third.

Zero page addressing modes take a one byte value as an operand to select the memory location. For example, if you want to load the contents of memory location $23 into the accumulator, you could enter the following:

```
LDA $23
```

Zero Page Indexed Addressing

Zero page indexed addressing uses the contents of the X register to determine the memory address to be accessed. Using the load accumulator instruction as an example, the memory address to be loaded is generated in the following way:

1. A memory address is specified in the instruction.
2. The value of the X register is added to this address.
3. The data from this generated address is loaded into the accumulator.

Zero page indexed addressing can only be used with the X register. Since the X register can contain an 8 bit value, an offset of up to $FF from the address specified in the instruction can be generated. Indexed addressing is used extensively when you are looking up a value in a table of data. A value corresponding to the distance into the lookup table would be loaded into the X register, then the indexed load instruction would be issued.

Absolute Addressing

Absolute addressing uses a two byte value in the operand to generate a 16 bit memory address. Due to this, the 6510 can address any byte in the range of $0000 to $FFFF. Normally, the assembler will select the most efficient version of this instruction. If it is possible to use zero page addressing instead of absolute addressing, the assembler will generate this form of the instruction.

Absolute Indexed Addressing

Absolute Indexed addressing works in the same way as zero page indexed addressing except that a two byte address is specified in the operand. Also, the Y register can be used as the index register in absolute indexed addressing. Using a load accumulator instruction, the following steps are taken to load a byte:

1. A memory address is specified in the instruction.
2. The value of the index register is added to this address.
3. The data from this generated address is loaded into the accumulator.

Indirect Addressing with Indexes

So far, all of the addressing modes have assumed that you knew where the data you were interested in was ahead of time. Since this is not always the case, there needs to be a way for the computer to determine an address during program execution.

Indirect addressing means that you are not telling the microprocessor where the data that you want to use is, but rather you are telling the microprocessor where it can find the address of the data that you want. The indirect address will always be stored in zero page memory in two consecutive memory locations. The lower order byte of the address is stored in the first memory location, and the high order byte of the address is stored in the next location. This gives a 16 bit address, so that the data can be anywhere in the microprocessor's normal address space.

When you give an indirect addressing command, the operand will be the address in zero page that contains the first byte of the address of the pair of memory locations where the appropriate address is stored. It is important to reserve enough space in zero page RAM to hold all of the indirect addresses that you may be generating.

At this point, indirect addressing should seem pretty easy to use, but there is a catch. The 6510 does not have true indirect addressing abilities for data movement or arithmetic instructions. Instead there are two subclasses of indirect addressing available. These are:

- Indexed Indirect X
- Indirect Indexed Y

We will look at the second one first because it is the most often used. As is implied by the name indirect indexed Y, this command is a combination of indirect addressing and indexed addressing. As

was mentioned earlier, the command will have the address of where the address may be found. After the microprocessor generates an address from the two zero page memory locations, the contents of the Y register is added to the address to form the final address. The data at this final address can then be accessed. If the value of the Y register is 0, the command will act like a true indirect addressing command. You must be sure that the Y register is set to the desired value before executing this command, or you will never be sure of where the data is coming from (or going to).

In indexed indirect X addressing, the value of the X register is added to the zero page address of where the indirect address can be found. This new zero page address is then used to generate the final address of where the data can be located. If the X register is set to zero, this command will act like a normal indirect addressing command. The normal use for this command is to use the X register as an index into a table of addresses located in zero page RAM.

Implied Addressing

Instructions that use implied addressing are only one byte long. Instead of having to give an address in the instruction, the microprocessor decides on which one of its internal registers to use based on the instruction. For instance, the TAX instruction will transfer the contents of the accumulator to the X register. This is known to the microprocessor without the need for any other addresses. Because the microprocessor doesn't need to calculate or load an address when using implied addressing, these instructions execute faster than any other type of instruction.

Relative Addressing

All of the branching instructions in the 6510 use the relative addressing mode. In this mode, instead of specifying an address for the destination of the branch, an offset from the current instruction to the destination of the branch it specifies in the operand. The offset is a one byte value. This gives a branch instruction the ability to branch over a range of +127 bytes to −127 bytes. Normally, you will use a label as the destination of the branch when writing your program, and the assembler will calculate the offset.

Indirect Addressing

There is only one instruction that uses indirect addressing in the 6510; it is an indirect jump. An indirect jump uses the principles of indirect addressing that were discussed earlier. The main difference between the indirect jump and the other indirect instructions is that a 2 byte value (16 bit) can be given as the address where the indirect data can be found. When using an indirect jump, the X and Y registers play no part in the address generation procedure.

Chapter 5
Organizing Your Program

Now that you have some understanding of what an assembler does and of the 6510 assembly language, you can begin to think about how to organize your program. All programs can be broken down as follows:

- Macro library
- System definitions
- RAM definitions
- Data definitions
- Main program
- Subroutines

Although it may not be obvious at this point, this is a very logical outline. The macro library must be assembled first. Quite often there will be macros that define data areas. A macro must be defined prior to its first use. There is no penalty in terms of storage space for a macro that is not used. The source code for a complete macro library is given in the file MACLIB, Listing C-1 in Appendix C. All of the macros in the file are described in detail in Appendix B. As they are given in the file, the macros will work with the Commodore Macro Assembler program. If you are using another assembler, you may have to modify them slightly before they can be used.

Once the macros are in the system, the machine's hardware registers should be defined. This is the starting point when you try to learn how a new computer works, as it forces you to become familiar with the hardware. A full set of system definitions can be found in the file SYSDEF, Listing C-2 in Appendix C. The names that are assigned to the various hardware registers by this file are used throughout the book, so it would be a good idea to refer to the listing at some point. Most of the registers will be described in detail in a later chapter.

The RAM that is to be used as variables for the program needs to be defined next. You will usually find that programming is easier if you define your variables before you start to write your program. These definitions do not have to be completed during the first sitting. As you progress in your program, you will find that you have not defined all of the RAM that you would like to use. Additions to

the RAM definitions tend to continue until the program is shipped or scrapped, whichever comes first.

After all the hardware registers and RAM that you are planning to use have been defined, you are ready to start entering data. Where you put the data is a matter of available space and personal preference. If you are going to put data immediately preceding the code, it would be a wise move to put a jump to the first instruction of your program before you define your first byte of data. In this way, you will always know what the starting address of your program is, no matter how the size of the data section may change. The data section will contain all the data that is not code.

This may seem like quite a few preliminaries to the actual program, but all of the steps do need to be taken. The actual source code that you will be creating will be making constant references to all of the names and definitions that have been defined previously.

When you are starting any program, especially one in assembly language, it is important to break the program into a number of smaller routines. Quite often, the small routines can be individually tested and later merged to form a complete program. Also, small segments can be saved for use later in other similar programs so that you won't have to start from scratch every time. Unless you write perfect programs every time, small program segments will be much easier to debug. If you write an entire program and then try to make it run, it can become quite difficult to determine where in the program the problem is. On the other hand, if you had tested all the small program segments before merging them into a complete program, the only problems that you might encounter should be in the interconnections between the program segments. Since you already know that all of the pieces of the program work, you should not have any difficulty finding the bugs.

As you are writing your program, do not be alarmed if you realize that you have not defined something you need to use. Simply write what you have forgotten on paper and use it in the code as if it had been defined. Then, when you feel like taking a break from the creative process, go back and insert your addition into the proper definition file. Until you try to assemble your program, the computer does not know or care what has or has not been defined.

You may have been wondering why the subroutines should be placed at the end of your program. Unlike the macro library, any subroutine in your program will use a certain amount of memory, whether it is used or not. Because of this, any subroutine that is not used should be deleted so as not to waste memory space or the time that it takes the assembler to assemble the subroutine.

By following this general outline, you will have a manageable and modular way with which to approach the design of your program. Most assemblers have a command that allows you to chain together different parts of your program. If your assembler has the ability, you may find it desirable to write all of the different modules of your program as separate files, and then let the assembler link them together as it assembles the program. This has some advantages over creating one massive file. For instance, if you need to add a definition to your RAM definitions, you only need to load the file with your other definitions. If your disk drive is particularly slow (as all Commodore 64 drives are), you will find a partitioning of your program quite a time saver. As your program progresses, the definitions and data areas will rarely need modifying. Keeping all of the parts of the program separate allows you to edit or print the part of the program that you are currently working on without having to deal with those parts of the program that have been tested and debugged.

Chapter 6
Working with Interrupts

In the Commodore 64 interrupts serve as a major source of timing and program control. The interrupts are normally used to maintain the real time clocks and the type ahead keyboard buffer. Interrupts can also be used to signal sprite collisions and inform the program when a specific scan line has been reached.

To best understand what an interrupt is, consider a normal program. The microprocessor reads its instructions one at a time and executes them in order. Whatever it is told to do first, is done first. If there is a certain condition that makes it necessary to perform a certain operation immediately, this condition must be repeatedly checked throughout the program to ensure it is taken care of promptly. For example, a collision between a bullet and a player sprite should immediately initiate an explosion sequence. In a normal program, you have to monitor the collision status register constantly and take appropriate action.

The alternative is to let the hardware check for the collision. When a collision occurs, the VIC II chip can send an interrupt request to the microprocessor. If interrupts are enabled, the processor will execute an indirect jump through location $FFFE when it finishes executing its current instruction. Location $FFFE usually points to a point in ROM that has an indirect jump instruction for a point in RAM. In the Commodore 64, the RAM location that ultimately will direct the jump is $0314. If the address of your routine to initiate the explosion sequence is placed in locations $0314 and $0315, this sequence will only be called on when a collision is detected.

Using an interrupt for this purpose relieves the main program of scanning the collision register constantly. Because of this, less of a burden is placed on the microprocessor during the main program.

Interrupts should be used for the part of your program that needs the highest priority in terms of microprocessor time. For instance, if you want to change the background color at a certain point on the screen, you need to use an interrupt. The VIC II chip can generate an interrupt on any scan line that you specify. This type of interrupt is called a *raster* interrupt. By using raster interrupts, your in-

terrupt routine can gain access to the processor at a specific (relatively) point on the screen.

Interrupts are useful when the main program involves lengthy calculations or is going to be busy for quite some time. If you are trying to maintain animation while the main program is running, you need to use some form of interrupt to take control at least once every other screen. Otherwise, the animation will appear jerky.

Before enabling your interrupt routine and thereby disabling the Commodore 64's operating system, you should disable all other sources of interrupts in the machine. Once this is done, you can always find the cause of the interrupt easily. Most of the functions performed by Commodore's operating system either are unnecessary in a game or can be done in a different manner.

The KILL macro essentially shuts down all of the devices in the system capable of generating interrupts. Once the interrupts from these chips have been disabled, you can safely change the interrupt vectors to point at your interrupt routine.

After the KILL macro is called, there are no interrupts generated in the system. This is a good time to change the addresses stored in the interrupt vectors. The nonmaskable interrupt vector (NMINV) should be changed to point at a return from interrupt (RTI) instruction. Nonmaskable interrupts are rarely, if ever, used in a game program. The maskable interrupt vector (CINV) should be changed to point at the first instruction of your interrupt routine.

Listing C-3 in Appendix C is the source code for a program that uses a RASTER interrupt to change the background color of the screen in the middle of the screen. This is also a good time to ensure that you are using your assembler properly. If you can successfully enter and run this short program, you should have no problem getting some of the longer programs to run later. The macro library and the system definitions that were defined earlier (Listings C-1 and C-2) are inserted into the program by the .LIB directive of the assembler. An executable version of this program is shown in Listing C-4.

To run the executable form of this program enter the following commands:

```
LOAD "DEMO.O",8,1
SYS 4096
```

After setting up the system, the main program just loops through itself. The only way that any changes can occur is through interrupts. The first interrupt is generated at the mid point in the screen, as specified by the first RAST macro. After changing the screen color to blue, this interrupt uses the RAST command to set an interrupt to occur at the bottom of the screen. Next, the address of the second interrupt is placed in the interrupt vector. The second interrupt (INT1) works in the same manner as the first, only it points to the first interrupt when it is done. By using two interrupt service routines in this manner, different things can be done on each half of the screen.

You may have noticed that you can see the point on the screen where the colors change, and it seems to be moving. This occurs because the microprocessor must finish the instruction it is currently working on before it can process the interrupt. Since an instruction may take from two to six machine cycles to execute, and the electron beam travels about three pixels per instruction cycle, the color can change in an 18 pixel area. This assumes that the VIC II chip is consistent about when it notifies the processor about the interrupt. Any timing inconsistencies in the VIC II chip enlarges the area where the color changes.

By adding a delay in the interrupt service routine before changing the background color, you can force the color change area to be in the border where it won't show.

After running this program, you can not reset the Commodore 64. By disabling the operating system, you have disabled the keyboard scan routines, effectively making the computer deaf to outside stimuli. When you have finished watching your new program and want to reset the machine, turn it off and back on again.

Warning: Always be sure that you have saved your program and source code before trying to run a new program!

Chapter 7
Technical Information

Up until now, this book has dealt with concepts and generalities when referring to the hardware in the Commodore 64. This was important to get you used to the capabilities of the hardware without bogging you down with details. This chapter will go into the details of getting the computer to generate the effects you are after. This chapter will probably be the chapter to which you will refer most often when writing a game program. All of the information that you will need in order to program the VIC II chips and the SID chip will be explained in this chapter. After you have become familiar with the hardware in the Commodore 64, you will not need the full explanation of the hardware registers. When you are up to such a level, you will find it easier to use the listing of the SYSDEF file in Appendix C as a quick reference guide to the registers.

Unless otherwise noted, all references to addresses in this chapter use hexadecimal notation. The names that have been assigned to the registers are the standard names that have been defined in the hardware definition listing, SYSDEF (Listing C-2 in Appendix C). By using this naming convention for the registers, you will begin to gain familiarity with the register names as they are used in the assembler. All of the names are made up of 6 characters or less, so no matter what type of assembler you are using, the same names will be acceptable.

Macros that supply many of the functions described below have been provided. Descriptions of all of the macros can be found in Appendix B, and Listing C-1 in Appendix C provides the source code. The macros may use a few more instructions to perform the function than would be required if you were to provide the data yourself. They do have advantages, however. Your program will be easier to understand if you use the macros, as there will be a recognizable name given to the macro as opposed to a sequence of assembly language instructions.

COMMODORE 64 ADDRESS SPACE

As you know, inside the Commodore 64 is a 6510 microprocessor, which controls the machine.

It uses a 16 bit address bus allowing it to access 2^16 or 65536 bytes of memory. This is all the memory that can be accessed at one time. Fortunately, as mentioned briefly before, through a technique called *bank switching*, different types of memory can be switched into or out of this address space. Through the use of bank switching, the 8K BASIC ROM can be accessed instead of 8K of RAM. The Commodore 64 switches other sections of RAM with ROM, and also switches an area of RAM with some hardware registers, such as the VIC II chip.

When choosing the appropriate memory map for your program, you must decide which of the functions provided by the Commodore 64 you will be using. For instance, if you do not need BASIC, you can switch the BASIC ROM out of the memory space. Doing so will give you 8K of RAM that you would not otherwise be able to use.

Some of the memory maps available to you will allow you to switch the 4K I/O space at $D000 with 4K of RAM. What this means is that you will no longer have access to the hardware registers at these locations. When it becomes necessary to change any of the values in one of these registers, you will need to switch the I/O space back into the RAM space. This is generally more trouble than it is worth, unless you desperately need the extra memory.

You also have the option of switching out the 8K KERNAL ROM. In most cases, you will not be using any of the KERNAL routines. If this is the case, there is no reason to keep it in memory. 8K of RAM can be switched into the space where the KERNAL ROM was.

Caution: All interrupts in the system should be shut down before the KERNAL ROM is switched out of memory. The six bytes of memory from $FFFA to $FFFF in the KERNAL contain the vectors for the interrupts. After the KERNAL has been switched out, new interrupt vectors need to be stored in RAM.

MEMORY CONTROL AND MAPPING

There are five control lines that control the bank switching of the different memory areas. Three of these lines are controlled by the microprocessor using its internal I/O port.

The three internal control lines are the least significant three bits at address $0001. This is the hardware I/O port of the 6510 processor. Data can be written to this port just as it can be written to any other memory location. Bits 0 and 1 are used to select from the four primary memory maps that are available. Bit 2 selects whether the I/O devices or the character generator ROM will be addressable by the microprocessor in the range of addresses from $D000 to $DFFF. Bit 2 has no effect on the system when the memory map with 64K of RAM accessible is selected. The remaining two control lines are internally pulled high when there is no cartridge plugged into the computer and can be ignored for a disk based program.

Figure 7-1 shows all of the memory mapping possibilities.

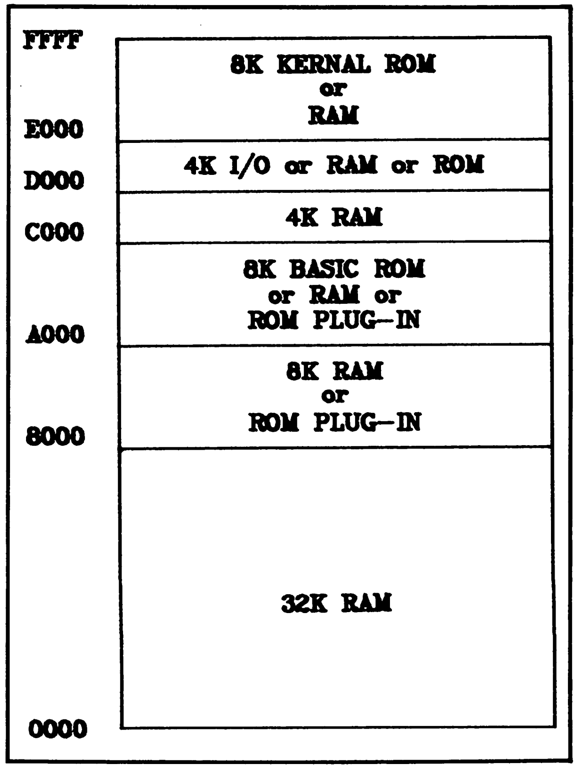


Fig. 7-1. Memory mapping options chart.

The memory maps in Figs 7-2, 7-3, 7-4, and 7-5 can be selected through software.

GRAPHICS MEMORY LOCATIONS

Although the Commodore 64 has 64K of memory, the VIC chip can only reference 16K of memory at any one time. Fortunately, you can change which of the four 16K blocks of memory in the computer the VIC chip will be able to use. When the Commodore 64 is powered up, bank 0 of memory ($0000-$3FFF) is selected. If you wish to change the bank of memory that the VIC chip will use, you must set the least significant two bits of $DD00 to the value that represents the desired bank. Before doing so, you must set bits 0 and 1 of $DD02 to 1. This will select the control bits that

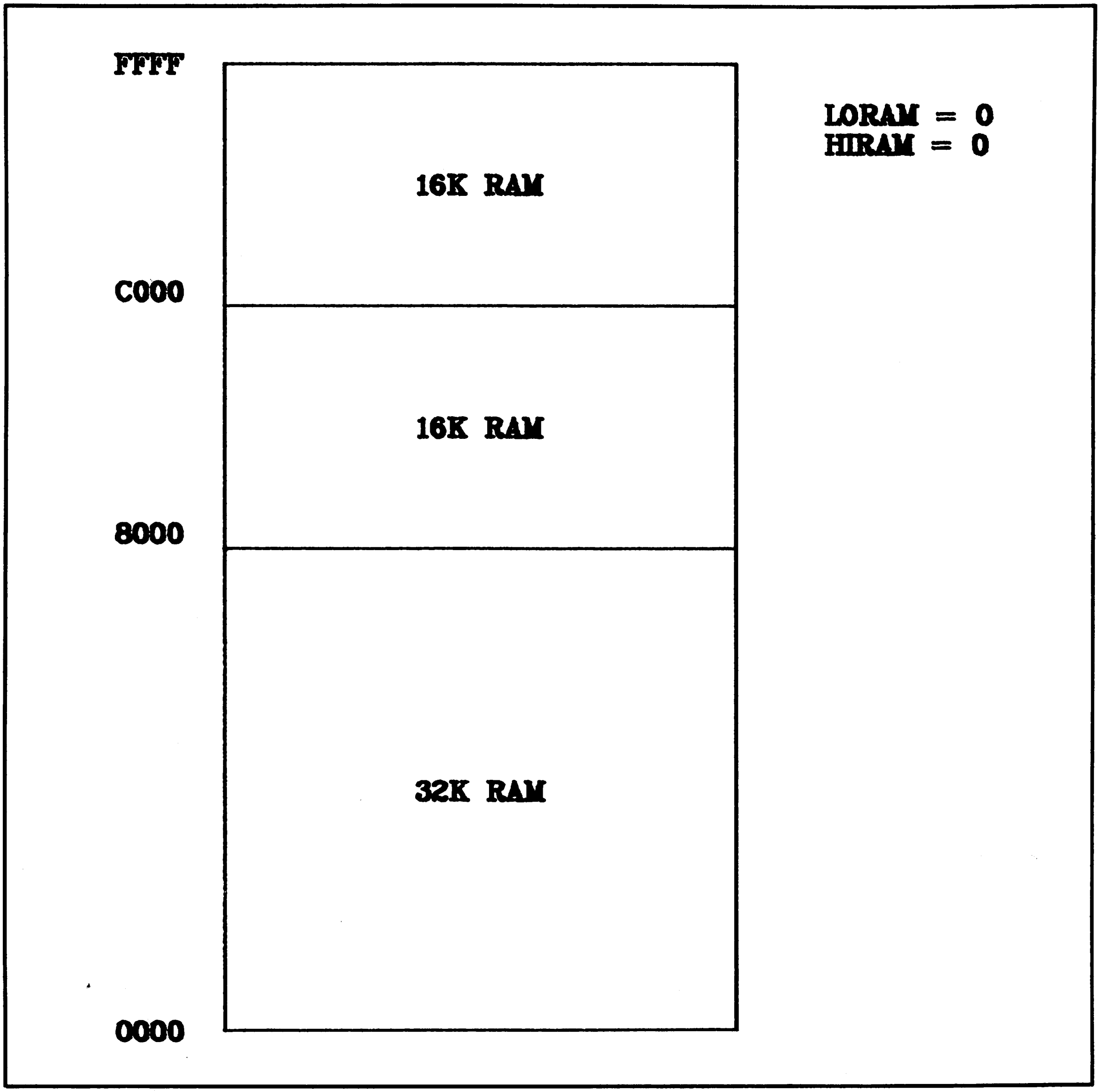


Fig. 7-2. 64K RAM memory map.

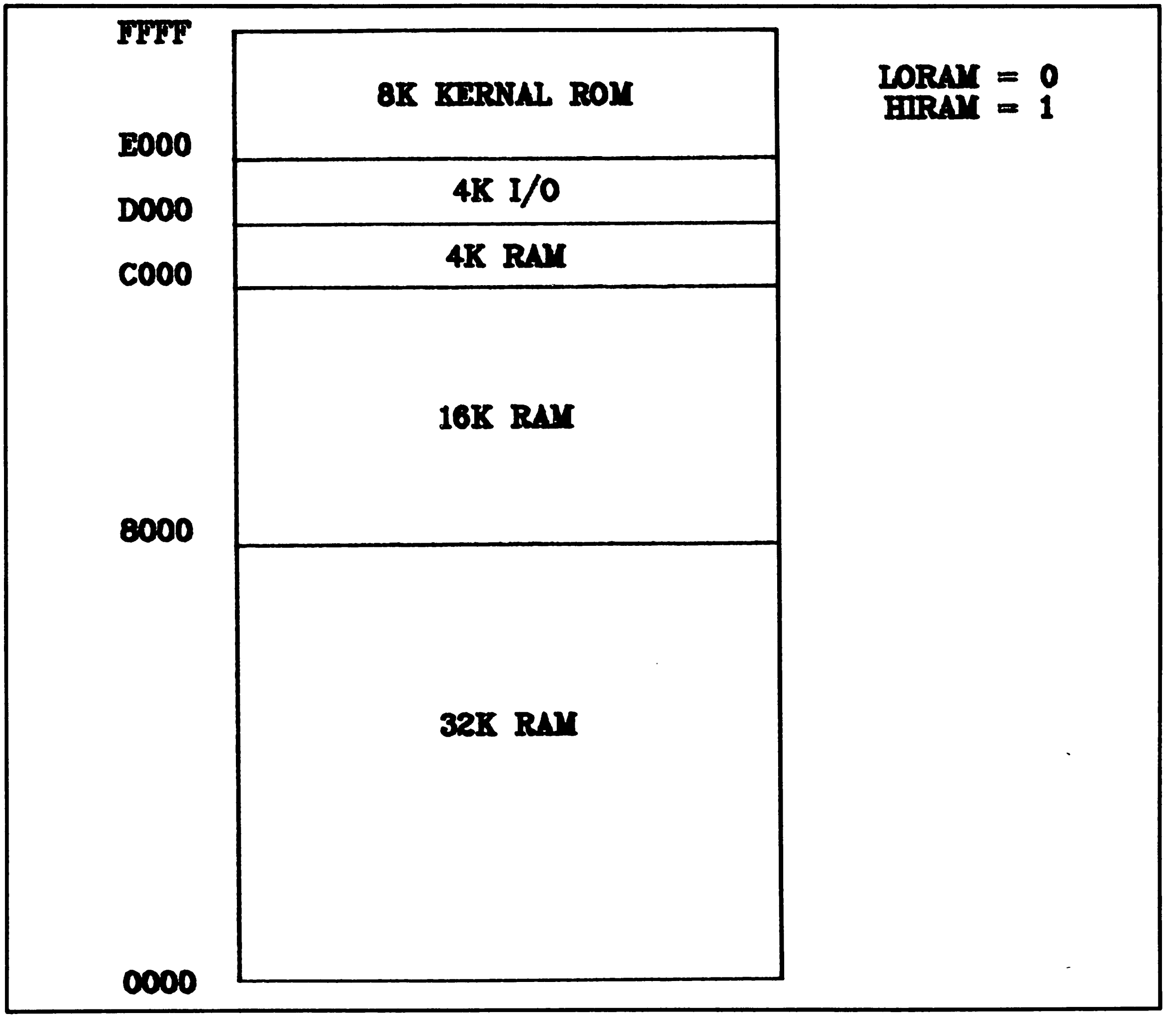


Fig. 7-3. 52K RAM memory map.

control-memory select to outputs. The BANK macro will select the proper bank for you. If you want to change the bank yourself, use the information in Table 7-1 for the appropriate values:

Table 7-1. The Values to Use When Selecting the Bank of Memory the VIC Chip Will Use.

VALUE	BANK #	MEMORY RANGE ADDRESSED
3	0	$0000-$3FFF
2	1	$4000-$7FFF
1	2	$8000-$BFFF
0	3	$C000-$FFFF

STANDARD TEXT MODE

When the Commodore 64 is first turned on, it owers up in its standard text mode. When in this mode, the screen is arranged as 40 characters by 25 lines. This gives a total of 1000 characters that can be displayed on the screen at any one time. The character to display in each position can be found in the 1000 bytes of RAM starting at $0400. If you want, you can instruct the VIC chip to use a different area of memory to find the text to be displayed. The upper four bits of the VIDBAS register ($D018) control where the VIC chip will

find the text. Text memory will always be found on $0400 boundaries.

Each byte in the text memory area is used as an index into a character generator section of memory. Since a byte can have 256 values, each position on the screen can display one of 256 patterns. When the Commodore 64 is powered up, the graphics information that the text memory references is in the character generator ROM. The VIC chip can be instructed to use a different area of memory as the character generator by changing the lower four bits of the VIDBAS register ($D018). This section of memory will be referred to as graphics memory, as the information can be of any type of graphics, not necessarily text.

The Commodore 64 has a built-in character

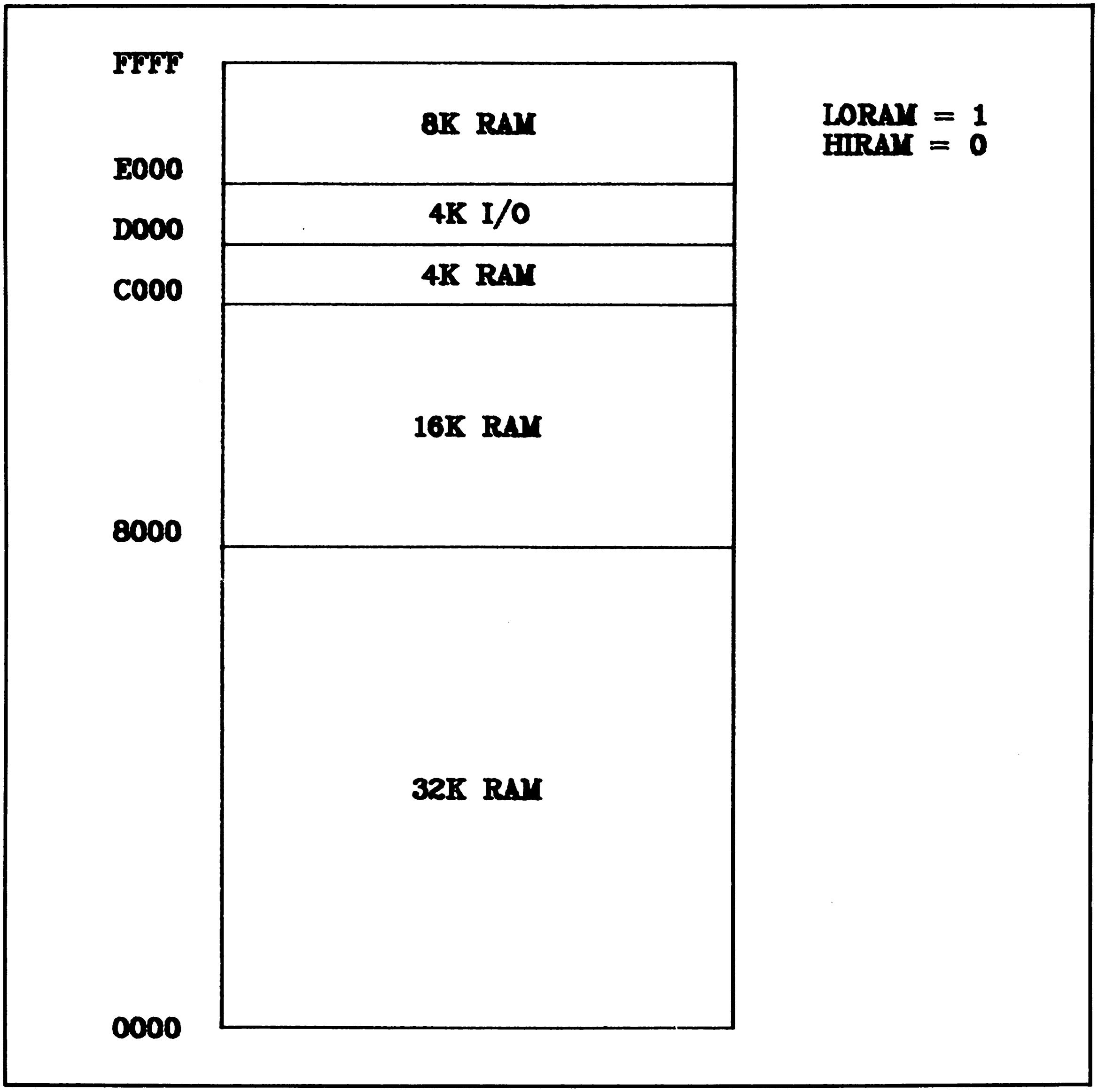


Fig. 7-4. 60K RAM memory map with no KERNAL ROM.

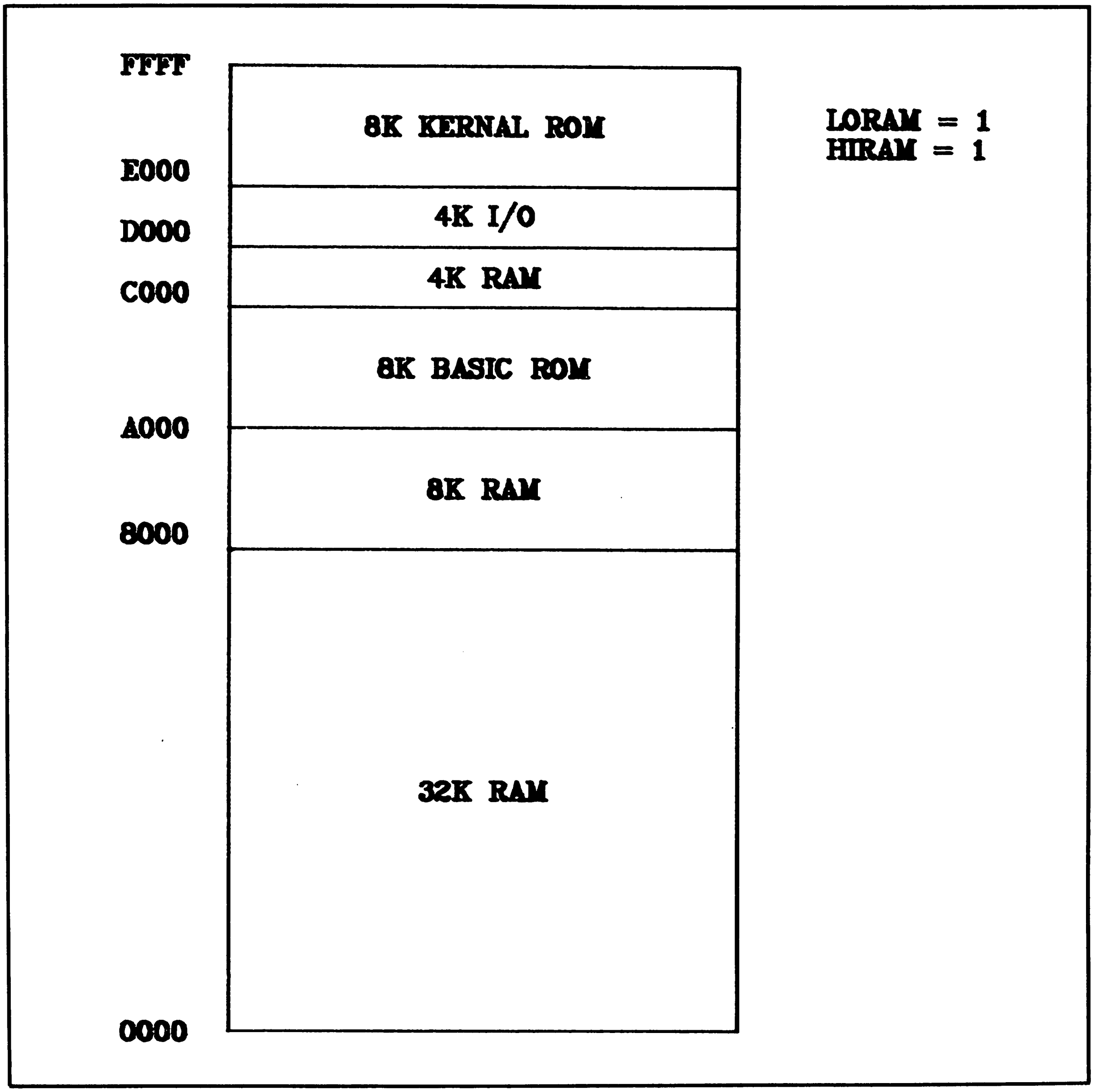


Fig. 7-5. The default memory map with 42K RAM.

generator ROM, which is physically at $D000-$DFFF. You may have noticed that this is the same address range where the hardware registers in the Commodore 64 are. This is possible only because the hardware registers are not available to the microprocessor at the same time as the character generate ROM is. Bit 2 of the I/O port at $01 controls whether the ROM or hardware registers will be available to the microprocessor. When this bit is set to a 1, the hardware registers will be available. When this bit is set to 0, the character generator ROM can be read by the microprocessor. The VIC chip does not see the character generator at this address, however. The VIC chip has been tricked into seeing the character generator from $1000-$1FFF. The VIC chip also

sees an image of the character generator ROM from $9000-$AFFF. Notice that the VIC chip can only see an image of the character generator when in banks 0 and 2 of memory. The microprocessor does not see the character generator in the same place that the VIC chip does, so there is no conflict with the microprocessor when using one of the areas in memory to store data that the VIC chip thinks is the character generator. You will normally choose either bank 1 or 3 for your graphics, so that the image of the character generator does not get in your way. If you do need to use text in your program, you can either copy data out of the character generator or create your own character set.

The TXBAS macro can be used to change the base address where the text will be located. Similarly, the GRABAS macro can be used to change the base address of graphics memory. It is important to remember that the text and graphics base addresses must be added to the bank that is currently selected in order to find the absolute address in memory where the data will be found. For example, if you had selected bank 1 of memory ($4000-$7FFF) and the text base address was $0400, the text data would be found at $4400. If you want to change the base address of the registers yourself, the values to use can be found in Table 7-2. When changing a value for the TEXT mode, only the upper bits should be changed in the VIDBAS register. Similarly, when changing a graphics base address, only the lower bits should be changed. A dash indicates a don't care condition. You must remember to add the BASE address register to the addresses to find the real address.

Table 7-2. The Values To Use To Change the Base Address for Text or Graphics.

VIDBAS VALUE Upper Bits	TEXT BASE ADDRESS
$0-	$0000
$1-	$0400
$2-	$0800
$3-	$0C00
$4-	$1000
$5-	$1400
$6-	$1800
$7-	$1C00
$8-	$2000
$9-	$2400
$A-	$2800
$B-	$2C00
$C-	$3000
$D-	$3400
$E-	$3800
$F-	$3C00

VIDBAS VALUE Lower Bits	GRAPHICS BASE ADDRESS
$-0	$0000
$-2	$0800
$-4	$1000 *
$-6	$1800 *
$-8	$2000
$-A	$2800
$-C	$3000
$-E	$3800

* In BANK 0 and 2, ROM images will appear here.

COLOR MEMORY

As discussed earlier, when in the text mode, the screen is arranged as 25 lines of 40 characters. Each of the characters can be one of 256 different patterns. In addition to being able to select the character to be displayed at each position on the screen, you can also select the color that you would like each character to be. The Commodore 64 provides 16 colors that can be used. There is an area in memory that is reserved to hold the color information. This section of memory is called the *color RAM*. Color RAM starts at $D800 and continues to $DBE7. Unlike the normal RAM used in the Commodore 64, color RAM is made up of nibbles rather than bytes. When you read a value out of color ram, only the least significant four bits are valid. You will read an eight bit value, but the data in the upper four bits is not predictable. Quite often, you must mask the upper four bits before using data from the color RAM.

Color memory does not move, and the VIC chip cannot use a different section of memory for the color information. The MVCOL macro will fill color

RAM with a single color. For instance, if you wanted all of the text to be white, you would enter "MVCOL WHITE." The text memory and color RAM are treated differently by the different graphics modes. If your picture doesn't look like you expect, check to be sure that the proper graphics mode has been selected.

CUSTOM CHARACTER SETS

Although the Commodore 64 has a built in character set, you will probably find that set limiting in what it allows you to do. The text mode can be used for quite a few different types of games if you redefine the character set. You can build up figures that are larger than one character size by placing the different pieces of the figure in adjacent locations. This gives you quite a bit of flexibility in using graphics as long as you define your own character set.

Each character is defined in memory as an array of 8 by 8 bits. This is stored as eight sequential bytes. Each byte has eight bits, and each bit represents one pixel on the screen. If a bit is on, the corresponding pixel on the screen will be turned on in the character color. If the bit is off, the background color is used for that pixel. The value in the text RAM is used as a lookup into the graphics RAM to choose which set of eight bytes will represent a character. If you decide to make your own character set, you will have to define any text characters that you may need as well as your graphics figures. Once you instruct the VIC chip to get its graphics data from RAM, it will no longer be able to use the character generator ROM.

Using a machine language monitor, you can enter the data for your character set into RAM. When you are finished, you should save your new set to the disk for later use. The first 8 bytes in your character set will be displayed if you place a 0 into a text memory location (assuming that you had previously instructed the VIC chip to get its graphics information from the place in memory where your new character set was placed). Your character set must begin on a multiple of $0800 in the current bank. The character set may not be placed in one of the locations where the VIC chip can see the character generator ROM.

MULTICOLOR MODE

To this point, all of the graphics in the text mode have been able to use only two colors for each character position on the screen. Normally a video game will use many more than two colors. To allow more colors to be displayed in a given area on the screen, the Commodore 64 has a multicolor mode that can be selected. In this mode, the character can use the character color and the colors in BCOL0 ($D021—the background color), BCOL1 ($D022), or BCOL2 ($D023). Using this mode, each pixel in a character location can be one of 4 colors. Unfortunately, you have to sacrifice 1/2 of the horizontal resolution to use this mode. This is usually a reasonable sacrifice considering how much more color can be used.

In order to turn on the multicolor mode, you must set bit 4 of the XSCRL® ($D012) or use the MULTON macro. To turn off the multicolor mode, you must clear bit 4 of the XSCRL register or use the MULTOF macro. In the text mode, the multicolor mode is selected individually for each character position on the screen. If the color in color RAM for a given character position is less than 8, that character position will be in the standard text mode. If the color in color RAM is 8 or greater, that character position will be in the multicolor mode. This allows you to mix standard and multicolor modes on the same screen.

As stated earlier, you sacrifice 1/2 of the horizontal resolution of the normal text mode when you select the multicolor mode. This means that each character will be made up of an array of 4 by 8 pixels. It still takes 8 bytes to define a character, but instead of each bit corresponding to a pixel on the screen, each PAIR of bits represents one of 4 registers to use to find the color for the bit pair. The following chart shows the correlation between bit pairs and registers.

Bit Pair	Register
00	BCOL0
01	BCOL1

Bit Pair	Register
10	BCOL3
11	Lower 3 bits of COLOR RAM

EXTENDED BACKGROUND COLOR MODE

In certain applications where you do not need a very large character set (less than 65 characters), you can use more colors than the standard text mode will allow by using the extended background color mode. In this mode, you do not sacrifice any resolution to gain extra colors; you only sacrifice the number of characters available from your character set. You can control both the foreground color (using the color RAM) and the background color of each character. The two most significant bits of the character code are used to select a background color from one of four registers, BCOL0 to BCOL3. Because of this, only the first 64 characters from your character set can be used.

The extended background color mode is enabled by setting bit 6 of the YSCRL ($D011) register. This mode can be turned off by clearing bit 6 of the YSCRL register. The following chart shows the relationship between the most significant two bits of the character code and the background color:

Bit Pair	Register
00	BCOL0
01	BCOL1
10	BCOL2
11	BCOL3

BIT MAPPING

Even though the various text modes available to you in the Commodore 64 provide many different options for displaying graphics, they still restrict you to using predefined shapes. For the majority of video game applications, you will be unable to use a text mode. The Commodore 64 has a high resolution bit mapped graphics mode that allows you to control each pixel on the screen individually. The display has a resolution of 320 pixels horizontally by 200 pixels vertically. This gives a total of 64000 pixels that can be controlled on the screen. Since each pixel is represented by a bit in graphics memory, 8000 bytes of graphics memory are required to represent the display.

There are two different types of bit mapped modes available on the Commodore 64. They are:

- Standard bitmapped mode: 320H by 200V, two colors per 8 by 8 group of pixels.
- Multicolor bitmapped mode: 160H by 200V, four colors per 8 by 8 group of pixels.

To turn on the standard bitmapped mode, you must set bit 5 of the YSCRL register. The GRAPH macro will perform this function for you. To turn off the bitmapped mode, you must clear bit 5 of the YSCRL register. You can use the TEXT macro to perform this function.

In the standard bitmapped mode, the colors for each 8 by 8 group of pixels are stored in text memory. The high 4 bits control the color of a pixel if its associated memory bit is on, while the lower 4 bits specify the color of a pixel if its bit is off. Color RAM is not used in the standard bitmapped mode. Table 7-3 shows how the bytes of graphics memory are organized with regard to the screen. This sequence is repeated for all 25 rows.

Table 7-3. The Bytes of Graphics Memory as Organized with Regard to the Screen.

ROW 0	0	$8	$10	$18	$138
	1	$9	$11	.	$139
	2	$A	$12	.	$13A
	3	$B	$13	.	$13B
	4	$C	$14	.	$13C
	5	$D	$15	.	$13D
	6	$E	$16	.	$13E
	7	$F	$17	.	$13F
ROW 1	$140	$148	$150	$158	$278
	$141	$149	$151	.	$279
	$142	$14A	$152	.	$27A
	$143	$14B	$153	.	$27B
	$144	$14C	$154	.	$27C
	$145	$14D	$155	.	$27D
	$146	$14E	$156	.	$27E
	$147	$14F	$157	.	$27F

Multicolor Bitmapped Mode

To turn on the multicolor bit mapped mode, you must first turn on the bit mapped graphics mode as shown above. Then you must set bit 5 of the YSCRL register. You can turn off the multicolor mode by clearing bit 5 of the YSCRL register. The macros MULTON and MULTOF can be used to turn on and off the multicolor modes.

In the multicolor mode, four colors can be displayed in each 4 by 8 group of pixels. Each byte in graphics memory is broken down into 4 bit pairs. Each bit pair specifies where the color information will be found for each pixel. In addition to the two colors that are defined in text memory, the color RAM is used in this mode to hold one more color. The fourth color is the background color that is stored in BCOL0 ($D021). The correspondence between the bit pairs in graphics RAM and where the color is found is shown in the following chart:

Bit Pair	Color
00	BCOL0
01	UPPER NIBBLE OF CHARACTER RAM
10	LOWER NIBBLE OF CHARACTER RAM
11	COLOR RAM

SPRITES

A *sprite* is a small moveable object block that can move independently of the background graphics. The VIC chip can display eight sprites on the screen at any one time. Sprites can be displayed on any one of the display modes, and they will look the same in all of them. You can have up to 256 different sprites defined at any one time, but only eight can be displayed at the same time. The sprite to be displayed can be changed by changing a one byte pointer, so animation can be easily performed by quickly switching through a few different sprite patterns. Sprites can be moved very smoothly by simply giving the VIC chip the X and Y coordinates of the upper left corner of the sprite.

Sprites have different display priorities. That means that the sprite with a higher priority will appear to move in front of a sprite with a lower priority. This can be used to give the illusion of three dimensional movement. The priority of a sprite to the background graphics is individually selected for each sprite. If the background is given priority, the sprite will appear to move behind the background graphics. For instance, if a tree was being displayed in the bit mapped graphics mode, and a sprite in the shape of a dog was to move past the tree, the dog would appear to be moving behind the tree.

Each sprite is a block 24 pixels horizontally by 21 pixels vertically. The pixels that are set to one use 1 of the 16 available colors. The pixels that are set to zero allow the background color to show through (are transparent). Like the other graphics modes, a sprite can be selected to be in the multicolor mode, giving it a resolution of 12 by 21 in three colors plus transparent. Wherever a sprite is transparent, whatever is behind the sprite will show through.

For those times when a larger sprite is necessary, the VIC chip has the option of doubling the horizontal size, the vertical size, or both. You will not increase the detail available in your sprite by using one of the multiply options, only the size. When a sprite is expanded, each of the pixels is twice the size of the pixels in a normal sprite.

Sprite Pointers

Once a sprite has been defined, the VIC chip needs to be told where to find the pattern. The sprite definition must be in the currently selected bank of memory for it to be displayed. Since each sprite definition takes up 64 bytes, a sprite definition will always start on a $0040 boundary in memory.

A 16K bank of memory can hold 256 sprite definitions, so it will only require one byte to tell the VIC chip which sprite to display. The sprite pointer is a number which, when multiplied by 64, will give the starting address of the sprite definition. Sprite definitions may not be placed in a section of memory where the VIC chip sees an image of the character generator ROM.

The VIC chip will read the eight sprite pointers from the last eight bytes of the 1K of text memory,

an offset of $03F8 from the text base address. Since only 1000 out of 1024 bytes of text memory are used to display characters on the screen, the sprite pointers will not interfere with screen graphics. For example, since the default setting of the text memory is at $0400, the first sprite pointer will be $07F8.

Sprite Controls

For most of the sprite control registers, each bit in the register corresponds to one of the sprites. For example, bit 0 represents sprite 0, bit 1 represents sprite 1, and so on. The rest of the sprite controls require a value (such as a vertical location), so there is one register for each sprite.

Enabling a sprite. Before a sprite can be seen, it must be enabled. The register SPREN ($D015), has an enable bit for each sprite. If the bit is set, the sprite will be enabled. The sprite will only be seen if the X and Y positions are set to the visible portion of the screen. A sprite can be disabled by clearing the appropriate bit.

Setting the sprite color. There are eight registers that are used to hold color information, one for each sprite. Any of the 16 available colors may be selected for each sprite. Each bit that is set in the sprite definition will cause a pixel to be displayed in the sprite color. If the bit is clear, the pixel will be transparent. The sprite color registers are:

Name	Address
SPRCL0	$D027
SPRCL1	$D028
SPRCL2	$D029
SPRCL3	$D02A
SPRCL4	$D02B
SPRCL5	$D02C
SPRCL6	$D02D
SPRCL7	$D02E

Setting the multicolor mode. The multicolor mode can be individually selected for each ($DO1C)/sprite by setting the appropriate bit in the MLTSP ($DO1C) register. Setting a bit will enable the multicolor mode, clearing the bit will disable the multicolor mode. When the multicolor mode is enabled, the horizontal resolution drops from 24 pixels across to 12 pixels. Each pair of bits in the sprite definition is treated as a bit pair, whose value determines which of the four colors will be selected for the pixel. Table 7-4 shows the relationship between the bit pairs and the color registers.

Table 7-4. The Relationship Between the Bit Pairs and the Color Registers in the Multicolor Mode.

Bit Pair	Description
00	TRANSPARENT, SCREEN COLOR
01	SPRITE MULTICOLOR REGISTER #0 ($D025)
10	SPRITE COLOR REGISTER
11	SPRITE MULTICOLOR REGISTER #1 ($D026)

Using the sprite multipliers. Each of the sprites can be expanded in either the X or Y direction. When a sprite is expanded, each pixel is displayed as twice the normal size in the direction of the expansion. The resolution of the sprite does not increase, only the size.

To expand a sprite in the X direction, the appropriate bit must be set in the SPRXSZ ($D01D) register. To return the sprite to its normal size, clear and bit.

The expansion of a sprite in the Y direction is done in the same way as the X expansion. You must set the appropriate bit in the SPRYSZ ($D017) register to expand the sprite. The sprite can be returned to its normal size by clearing its bit in the SPRYSZ register. The sprite can also be expanded in both the X and Y directions by setting its bit in both registers.

Positioning sprites. Each sprite can be positioned independently anywhere on the visible screen and off the visible screen in any direction. Since the screen is 320 pixels wide, it takes more than one byte to specify a horizontal position. Each sprite has its own X position register and Y position register, and a bit in an extra most significant bit register. These registers are shown in Table 7-5.

The location specified by the registers is the position where the upper left corner of the sprite will appear.

Table 7-5. The Position Registers for the Sprites.

Address	Name	Description
$D000	SPR0X	SPRITE 0 HORIZONTAL
$D001	SPR0Y	SPRITE 0 VERTICAL
$D002	SPR1X	SPRITE 1 HORIZONTAL
$D003	SPR1Y	SPRITE 1 VERTICAL
$D004	SPR2X	SPRITE 2 HORIZONTAL
$D005	SPR2Y	SPRITE 2 VERTICAL
$D006	SPR3X	SPRITE 3 HORIZONTAL
$D007	SPR3Y	SPRITE 3 VERTICAL
$D008	SPR4X	SPRITE 4 HORIZONTAL
$D009	SPR4Y	SPRITE 4 VERTICAL
$D00A	SPR5X	SPRITE 5 HORIZONTAL
$D00B	SPR5Y	SPRITE 5 VERTICAL
$D00C	SPR6X	SPRITE 6 HORIZONTAL
$D00D	SPR6Y	SPRITE 6 VERTICAL
$D00E	SPR7X	SPRITE 7 HORIZONTAL
$D00F	SPR7Y	SPRITE 7 VERTICAL
$D010	XMSB	MOST SIGNIFICANT BIT REGISTER

The value placed in the Y position register will specify the vertical position of the sprite on the screen. This value may be up to 255. For an unexpanded sprite to be completely visible, the Y value must be between $32 and $E9. Any other values will place the sprite partially off the screen.

Whatever value is placed in the X position register is the least significant 8 bits of a 9 bit value. Each sprite has a ninth bit in the XMSB ($D010) register. An unexpanded sprite will be completely visible if the 9 bit X value is greater than $18 and less than $140. The HINC and HDEC macros can be used to perform 9 bit increments and decrements of the X position.

Table 7-6 shows the screen coordinates for expanded and unexpanded sprites to be fully visible on the screen. Any sprite positions outside of these limits will be partially or fully off of the screen. This provides an easy way to reveal a sprite gradually.

Assigning sprite priorities. As mentioned before, each sprite has a display priority with

Table 7-6. The Screen Coordinates at which Normal and Expanded Sprites Will Be Fully Visible.

POSITION	X	Y	X EXP	Y EXP
UPPER LEFT	$18	$32	$18	$32
UPPER RIGHT	$140	$32	$128	$32
LOWER LEFT	$18	$E5	$18	$D0
LOWER RIGHT	$140	$E5	$128	$D0

respect to the other sprites and to the background. You can create a three dimensional effect by allowing different sprites to pass in front of each other. The priority of one sprite to another is predetermined by the VIC chip. Sprite 0 has the highest priority, meaning that it will appear to be in front of all other sprites. Sprite 7 has the lowest priority of all the sprites.

Each sprite can be individually selected to either have a higher priority than the background or a lower priority. If the sprite's bit in the BPRIOR ($D01B) register is clear, the sprite will appear to pass in front of the background. When the bit for the sprite is set in the BPRIOR register, the sprite will appear to move behind the background image and in front of the background color. Because sprites can have transparent as one of their colors, any sprite that passes behind a higher priority sprite with transparent in it will show through in the transparent areas.

COLLISION DETECTION

The VIC chip can detect collisions between sprites and also between a sprite and the background. The VIC chip will defect collisions between the nontransparent portions of sprites.

When a collision between two sprites occurs, their bits are set in the SSCOL ($D01E) register. The data in the SSCOL register will stay valid until the byte is read. After the register is read, the data will be cleared, so it is important to store the data somewhere before analyzing it. The VIC chip will detect a collision even if the sprites are off the screen. One thing that should be noted is that the SSCOL register will only tell you which sprites are

involved in collisions, not which sprite hit which sprite. If you are multiplexing sprites, the data in the SSCOL register may be useless.

Sprite to background collisions are handled in almost the same way. The SBCOL ($D01F) register will detect a collision between the nontransparent portion of a sprite and the background. In a multicolor screen mode, the bit pair 01 is considered transparent for collision detection. Like the SSCOL register, the data is cleared after reading it.

BLANKING THE SCREEN

The entire screen can be blanked to the border color by clearing bit 4 of the YSCRL register. The screen can be turned back on by setting bit 4 of the YSCRL register. Blanking the screen does not disrupt any data on the screen. When the screen is blanked, your program will run slightly faster because the VIC chip doesn't need to fetch any data from memory.

THE RASTER REGISTER

The VIC chip keeps track of which scan line the electron beam is currently on. Since there are more than 255 scan lines in one TV frame, this will be a 9 bit value. The least significant 8 bits of the current scan line can be read by reading the RASTER ($D012) register. The ninth bit can found in bit 7 of the YSCRL register.

You can write a 9 bit value to the RASTER register and bit 7 of the YSCRL register. When the scan line reaches the value that you stored, bit 0 of the VIRQ ($D019) register will be set. If bit 0 of the VIRQM ($D01A) register is set, an interrupt will be sent to the microprocessor. You must remember to store a ninth bit when storing a RASTER number, or the comparison will not take place. The RAST macro will set the 9 bit raster number for you.

VIDEO INTERRUPTS

Different conditions within the VIC chip can generate interrupts. The interrupt status can be read by reading the video interrupts register, VIRQ ($D019). The bits have the following meanings:

Bit	**Type of Interrupt**
0	RASTER
1	SPRITE TO BACKGROUND COLLISION
2	SPRITE TO SPRITE COLLISION
3	LIGHT PEN
7	SET ON ANY ENABLED INTERRUPT

Once an interrupt bit has been set, a 1 must be written to that bit position in order to clear it. This allows you to process interrupts one at a time, without having to store the data elsewhere.

Interrupts will only be sent to the microprocessor if the corresponding bit in the *video interrupt mask* register, VIRQM ($D01A), is set. You will still be able to read the interrupts from the VIRQ register, but if the appropriate bit in the VIRQM register is not set, no interrupts will be generated. See the section on using interrupts for more information on using interrupts properly.

SCROLLING

One of the most advanced features of the VIC chip is its ability to smoothly scroll the screen in either the X or Y direction. The VIC chip can scroll the screen using hardware, freeing the microprocessor from the task of finely scrolling the screen. When the screen needs to be scrolled, the VIC chip can be instructed to scroll the screen within a range of 8 pixels in the X direction, the Y direction, or both.

The least significant three bits of the YSCRL ($D011) register control the amount of vertical scrolling. Since this register is also used for a number of control functions, the register should be read before it is changed. The XSCRL ($D016) register works in the same way as the YSCRL register except that the XSCRL register controls the amount of horizontal scrolling. When changing either of these registers, the lower 3 bits should be masked to 0, and the number of pixels to be scrolled should be OR'd to the new value. The result of this procedure can then be stored back into the register.

The following is a routine that can be used to change the value of the YSCRL or XSCRL register. This example shows how to set the YSCRL register to a scroll value of 7 without disturbing the value of the upper bits of the register.

```
LDA    YSCRL    ;LOAD THE DATA
AND    #$F8     ;MASK THE LOWER 3 BITS
ORA    #$07     ;SCROLL 7 PIXELS
STA    YSCRL    ;STORE THE NEW VALUE
```

As the scrolling value goes from 0 to 7 in the YSCRL register, the screen will scroll down. As the value in the XSCRL register goes from 0 to 7, the screen will scroll to the right.

When scrolling the screen, you will usually want to expand the border area of the screen. This will give you an area to place the new graphics to be scrolled onto the screen where they will not be seen. The VIC chip has two controls that will expand the border. The first of these is a 38 column mode. This mode can be selected by clearing bit 3 of the XSCRL register. The VIC chip can be returned to the 40 column mode by setting bit 3 of the XSCRL register. In the 38 column mode, one column on the right side of the screen and one column from the left side of the screen are covered by the border color. This will give you a buffer area where changes to the screen will not be seen.

The other border expansion option is useful for vertical scrolling. By clearing bit 3 of the YSCRL register, when the vertical scroll is set to 3, half of the top row and half of the bottom row will be covered by the border. The VIC chip can be returned to the normal 25 row mode by setting bit 3 of the YSCRL register. When the vertical scroll is set to 0, the top line will be entirely covered by the border. When the vertical scroll is set to 7, the bottom line of the screen will be entirely covered by the border.

Once you have reached a maximum scroll value in the X or Y direction, you will have to shift each character on the screen in the direction of the scroll in order to continue scrolling. After moving all of the characters on the screen, you can reset the fine scrolling registers to their minimum value and continue to use the hardware registers to scroll the screen.

There are a number of things that must be taken into account when writing a scrolling program. In order for the screen to appear to be in continuous smooth motion, the routine that will shift each character must be extremely fast. Also, if you are using a number of different colors in color RAM, each character in color RAM must be moved in the direction of the scroll at the same time as the characters in screen memory. If you do not need to scroll the entire screen, your program can be much shorter and will run faster. If your program does not run fast enough, you will see breaks in your graphics where the characters that have been scrolled are adjacent to characters that have not yet been scrolled.

If possible, your routine should be fast enough to reposition the entire screen in one screen update time (1/60 of a second). You can get by with a slower routine if you scroll your screen memory into a different area of memory than the one that is currently being displayed. This must be completed before the fine scrolling register reaches its limit, so when the entire screen needs to be repositioned, it will be ready. Instead of repositioning the entire screen at that point, which would have to be done within 1/60 of a second, all you need to do is to use the TXBAS or GRABAS macro to instruct the VIC chip to get the data from the area that has already been repositioned. The macro that you will use depends on the graphics mode that the screen is in.

JOYSTICKS

At some point, you will want to allow the player to have control over his character in the game. The most common form of input to a video game is a joystick. The Commodore 64 has two input ports that can be used for joysticks. By setting both DDRA ($DC02) and DDRB ($DC03) to $00, the two ports will be configured as inputs. Once the ports have been configured, the data from the joysticks can be read from JOY1 ($DC00) or JOY2 ($DC01). Bit 4 of a joystick port represents the fire button on that joystick. If that bit is clear, the fire button

is depressed. The lower five bits of a joystick port represent the direction of the joystick as shown below:

Bit	Direction
0	UP
1	DOWN
2	LEFT
3	RIGHT
4	FIRE BUTTON

When a contact on the joystick is pressed, its corresponding bit in the joystick register is clear. When two out of the lower four bits are clear, the joystick is on an angle. If you find it more convenient or manageable to have a bit set representing a closed switch, the NOT macro can be used to invert the data. Before using the joystick data as any type of an index into a table of data, you must mask the unused bits in the register. You will normally want to mask the entire upper nibble and treat the fire button separately. The following routine will invert the joystick data from port 0 and mask the unused bits as well as the fire bit. The result will be a four bit value that represents the joystick direction.

```
LDA JOY1        ;READ THE JOYSTICK PORT
NOT             ;COMPLEMENT THE
                ;ACCUMULATOR
AND #$0F        ;MASK THE UPPER BITS TO 0
```

Chapter 8
Sound Effects

One of the most advanced features of the Commodore 64 is its ability to generate sounds. Built into the Commodore 64 is a highly advanced sound generator, the sound interface device or SID chip. This chip has the capability of generating three independent tones over a range of more than six octaves. It has controls in each channel to control the attack, decay, and release times, and a sustain level for the volume of each channel. In fact, most of the features found in a musical synthesizer can be found in the SID chip.

Figures 8-1, 8-2, 8-3, and 8-4 show some of the different waveforms that the SID chip can generate. Figure 8-1 shows a triangular waveform, which will produce the cleanest tone, as it is a fairly close approximation of a pure sine wave. Figure 8-2 shows a sawtooth waveform. You will notice that it has sharper edges to it than the triangular waveform. Sharp edges on a waveform tend to generate various harmonics of the tone, so in general, the sharper the edges in a waveform, the more harmonics will be generated. The differences in the harmonic content can be heard as a difference in harshness of the tone. A triangular waveform will produce a very smooth, soft tone, while the square waveform, as shown in Fig. 8-3, produces a very sharp biting tone. When you select the square waveform in the SID chip, you have extra control over the tone of the sound. You can program in a pulsewidth for the wave, which varies the symmetry of the square wave. Depending on the settings that you use, the waveform will be more rectangular than square, as shown in Fig. 8-4.

The SID chip can also generate a noise waveform. This is a random signal that changes at the oscillator frequency. Many games will use this waveform to generate explosions, wind storms, or any type of sound that is not a specific tone. In addition, if channel 3 is set to generate noise, the amplitude of the waveform can be read by the microprocessor at any time. Because this is a constantly changing value, the number read will be a random number.

FILTERING

After you have created a sound, you can change

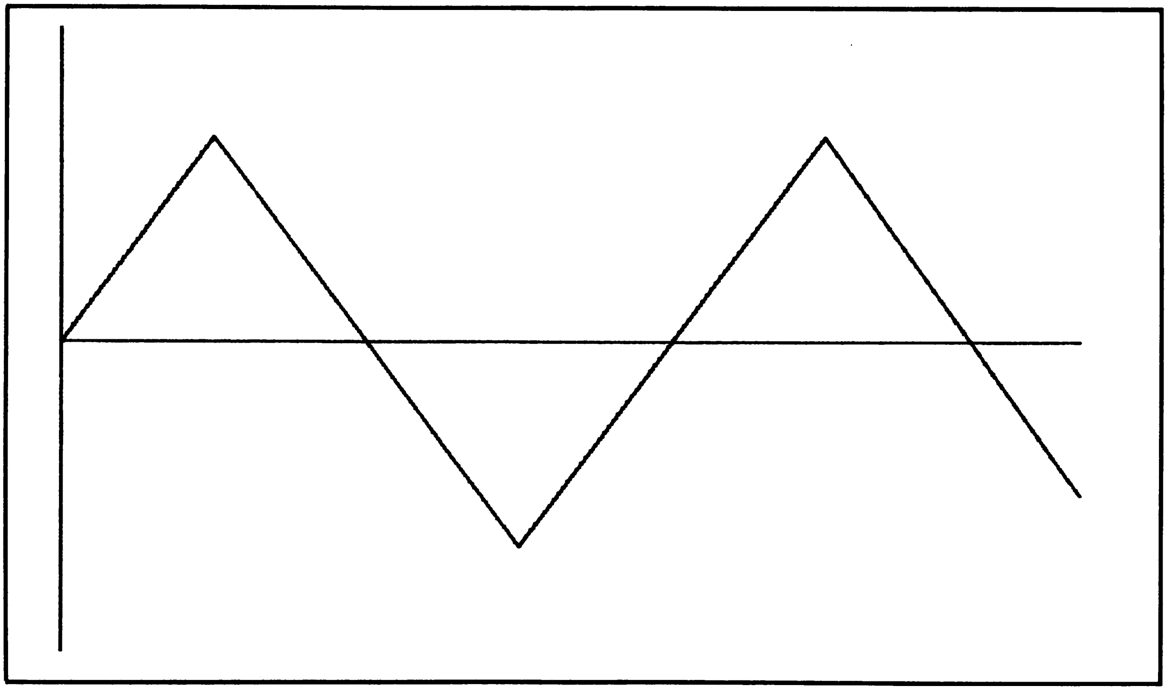


Fig. 8-1. Triangular waveform.

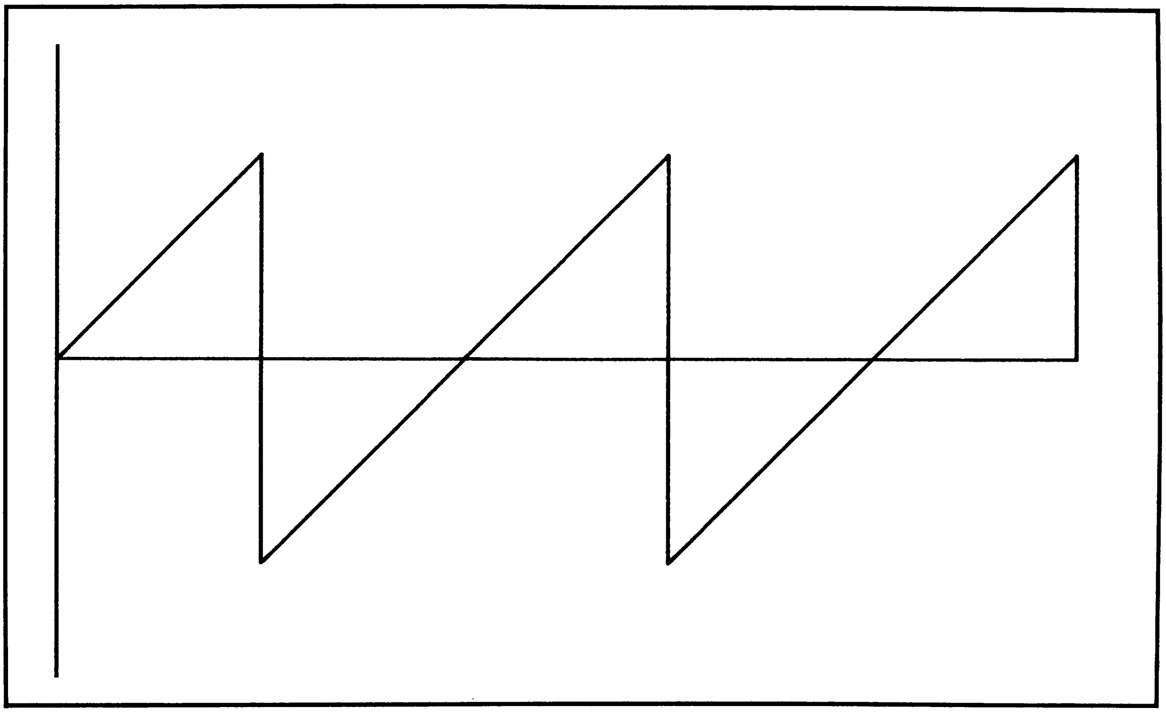


Fig. 8-2. Sawtooth waveform.

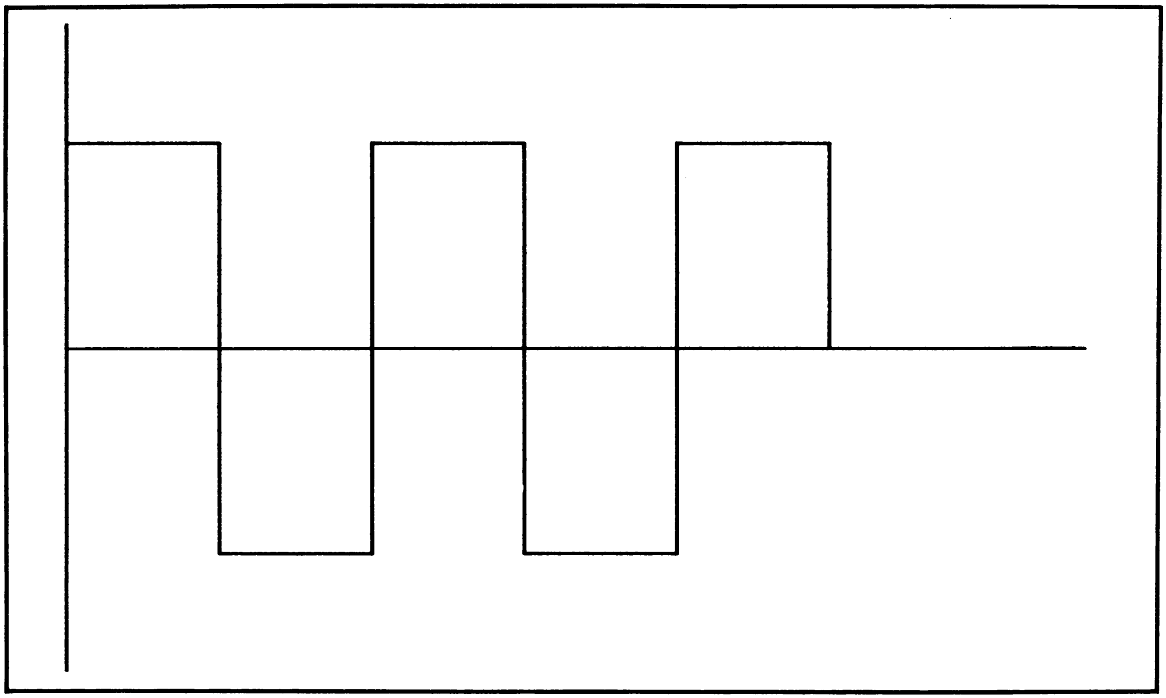


Fig. 8-3. Square waveform.

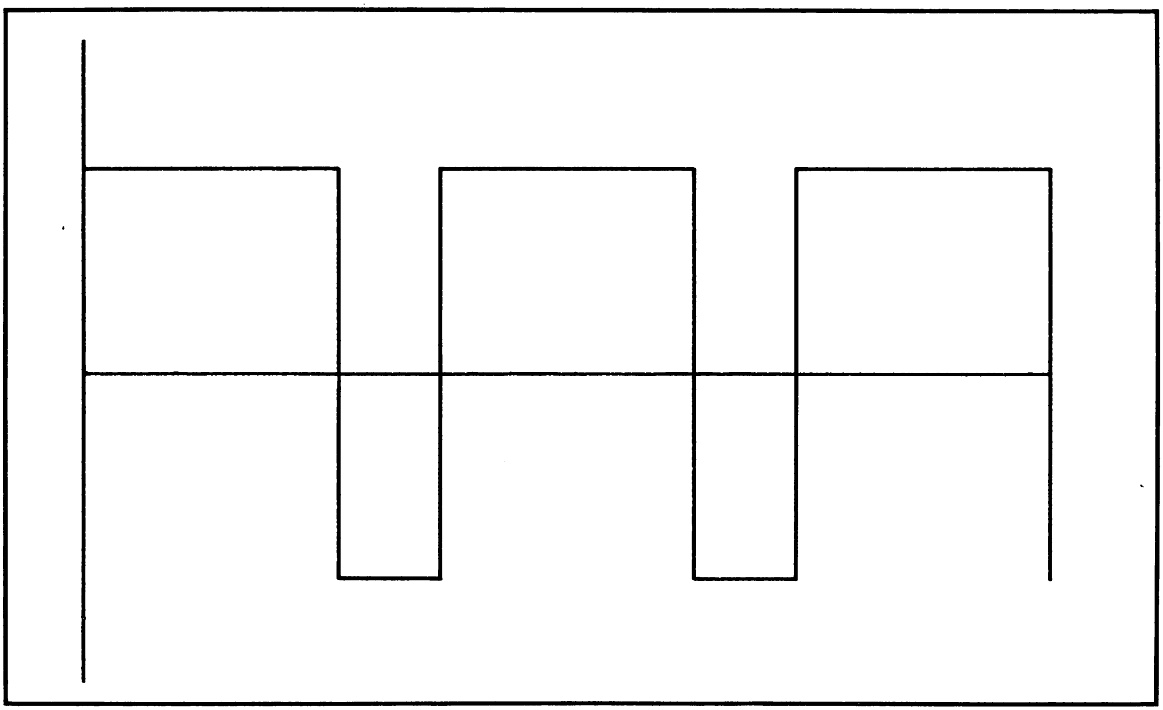


Fig. 8-4. Rectangular waveform.

it drastically by having the sound pass through one or more types of filters. An audio filter changes the sound by cutting down the volume of certain frequencies. Different types of filters will modify a sound in different ways. Each channel can be individually selected as to whether or not it will be passed through the filter.

The ability to route the audio outputs through one or more filters is a very powerful feature of the Commodore 64. Unlike most computers, which allow you to generate only simple tones, the filtering modes of the SID chip allow you to generate complex tones by modifying the harmonic content of the tones. The filters accomplish this task through a technique known as subtractive synthesis. By using an input source that is high in harmonics, the filter can selectively eliminate specific frequencies. Depending on the filtering mode, the same initial tone can be used to create many different sounds. In addition to using the filtering modes in a static fashion (setting them and leaving them), you can control the filter settings in real time. By doing so, you will be able to create sounds such as wind storms and jet engines. These are sounds that cannot normally be generated well by a home computer.

There are three types of filters in the SID chip, a low pass filter, a band pass filter, and a high pass filter. More than one type of filter can be selected at one time. When multiple filters are selected, the effects are additive. A notch filter can be created by selecting both the low pass and high pass filters. The filters will affect the sound in the following ways.

Low pass filter. When the low pass filter is selected, all frequencies above the cutoff frequency are attenuated at the rate of 12 dB/Octave. This filtering mode will generate a full sound.

Band pass filter. The bandpass filter will attenuate all of the frequencies above and below the cutoff frequency at the rate of 6 dB/Octave. A bandpass filter produces thin sounds.

High pass filter. All of the frequencies below the cutoff frequency will be attenuated at the rate of 12 dB/Octave when the highpass filter is selected. Tinny sounds can be generated when using this mode. Figures 8-5 through 8-7 show graphics fre-

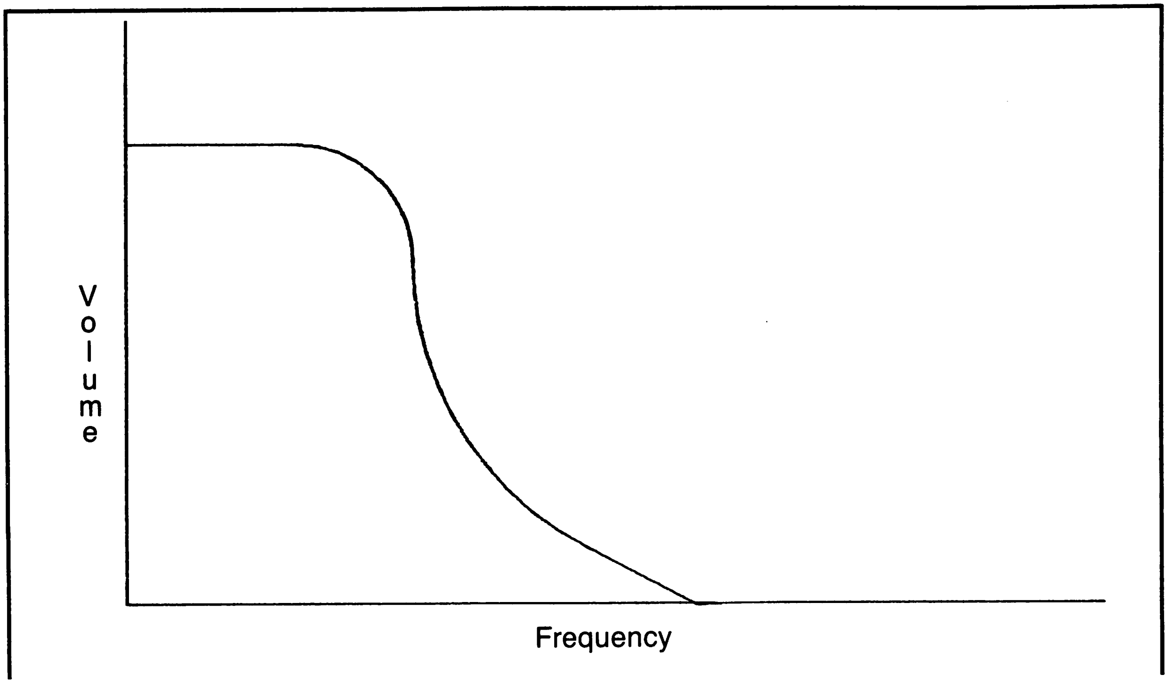
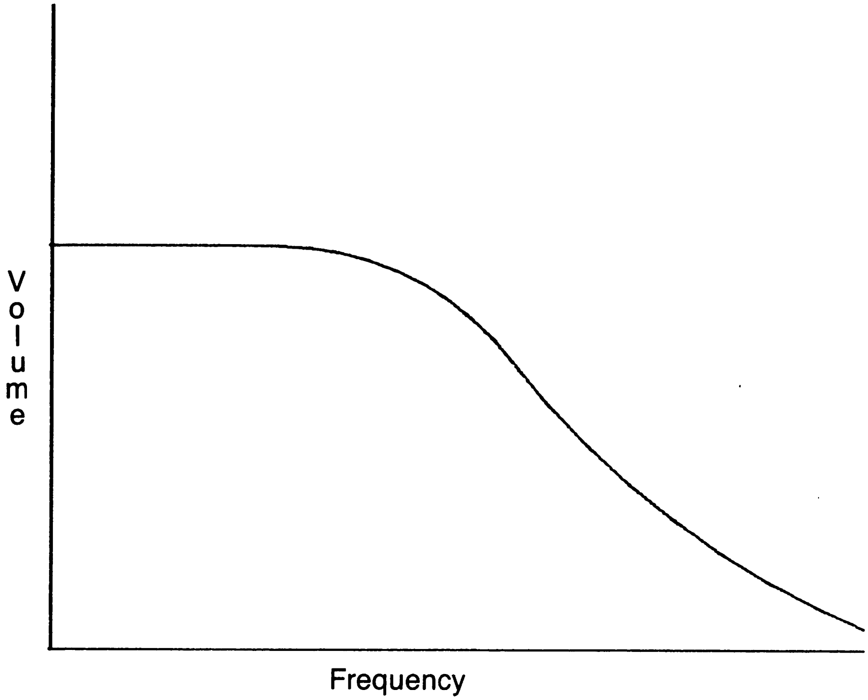
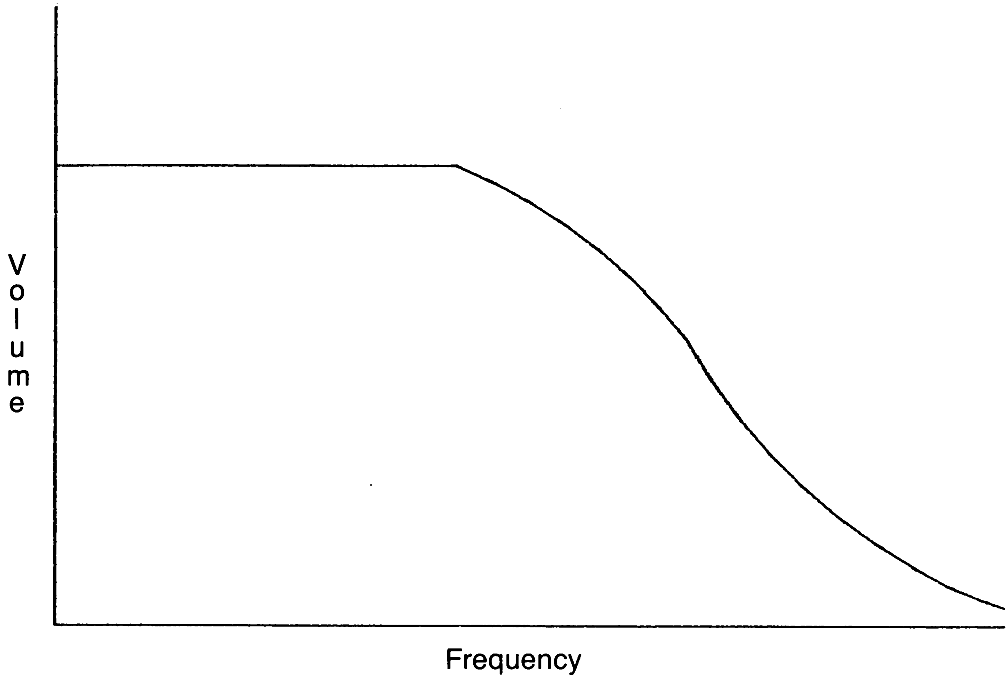


Fig. 8-5. The effects of changing the filter cutoff frequency for a lo-pass filter (continued on p. 40).


Volume
Frequency


Volume
Frequency

quency amplitude as the cutoff frequency of the filters are changed.

THE SOUND GENERATOR DEMO

Included in Appendix C is a program that shows many of the abilities of the SID chip. Listing C-5 is the source code; listing C-6 is the assembled code. This program loads three machine language files COMMON in Listing C-7, SNDDEF, in Listing C-8, and DATA in Listing C-9. By listening to this demo, you will begin to hear some of the possible sounds that can be created on the Commodore 64. As the demo is running, it will show you on the screen what effect it is currently demonstrating. Some of the longer attack and decay times will take a while to demonstrate, so please be patient. To run the demo, type in the following after checking to make sure the volume on your monitor is turned up:

```
LOAD"SOUND DEMO",8,1
SYS 4096
```

The sounds and effects in the program are by no means all of the sound effects that can be created by the SID chip. Depending on how sophisticated you care to become, you will be able to get many effects that are not directly obvious. For instance, if you write a program that changes the volume of one of the channels in real time, you will have full control over generating different types of tremelo effects.

There are program segments in the demo that can be used directly or expanded to help generate almost any type of sound or tune. The routine that generates the short tune that is played to show the effects of different waveforms simply reads a list of notes and note times. If the value of a note is $00, then the routine will quit. By changing the note values in the note table and the time values in the time table, this routine will play any type of tune.

THE SOUND EDITOR

In order to aid you in choosing values for the different registers, in the SID chip, a sound editing program has been included in Appendix C. Listing

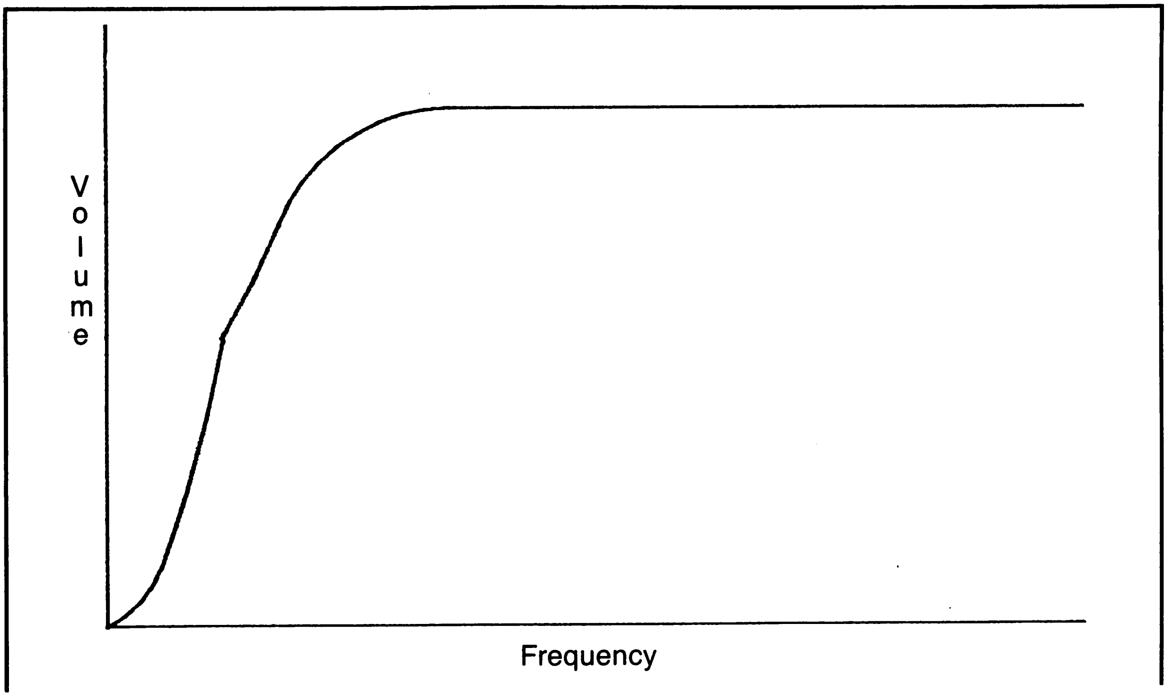
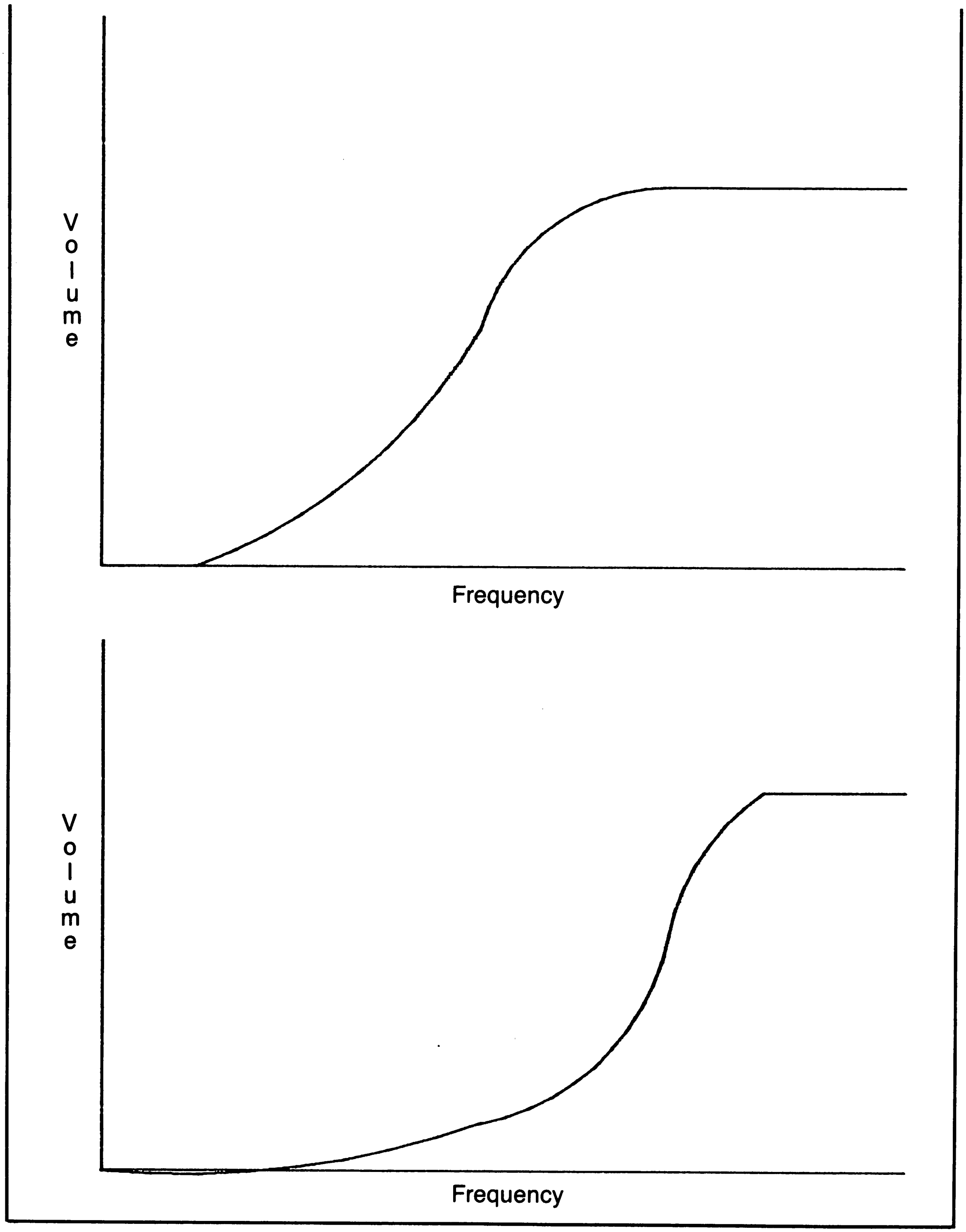


Fig. 8-6. The effects of changing the filter cutoff frequency for a hi-pass filter (continued on p. 42).


Volume
Frequency
Volume
Frequency

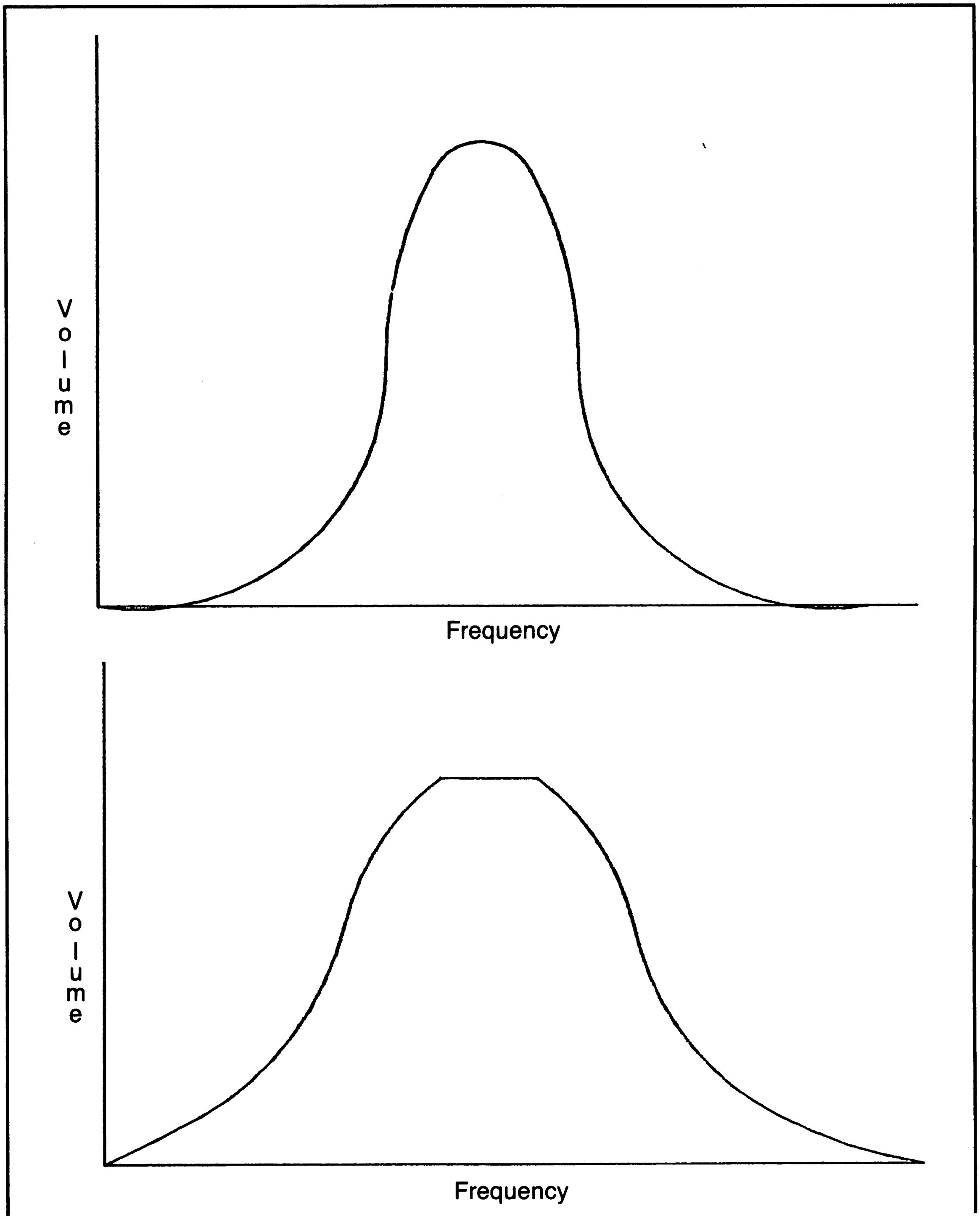
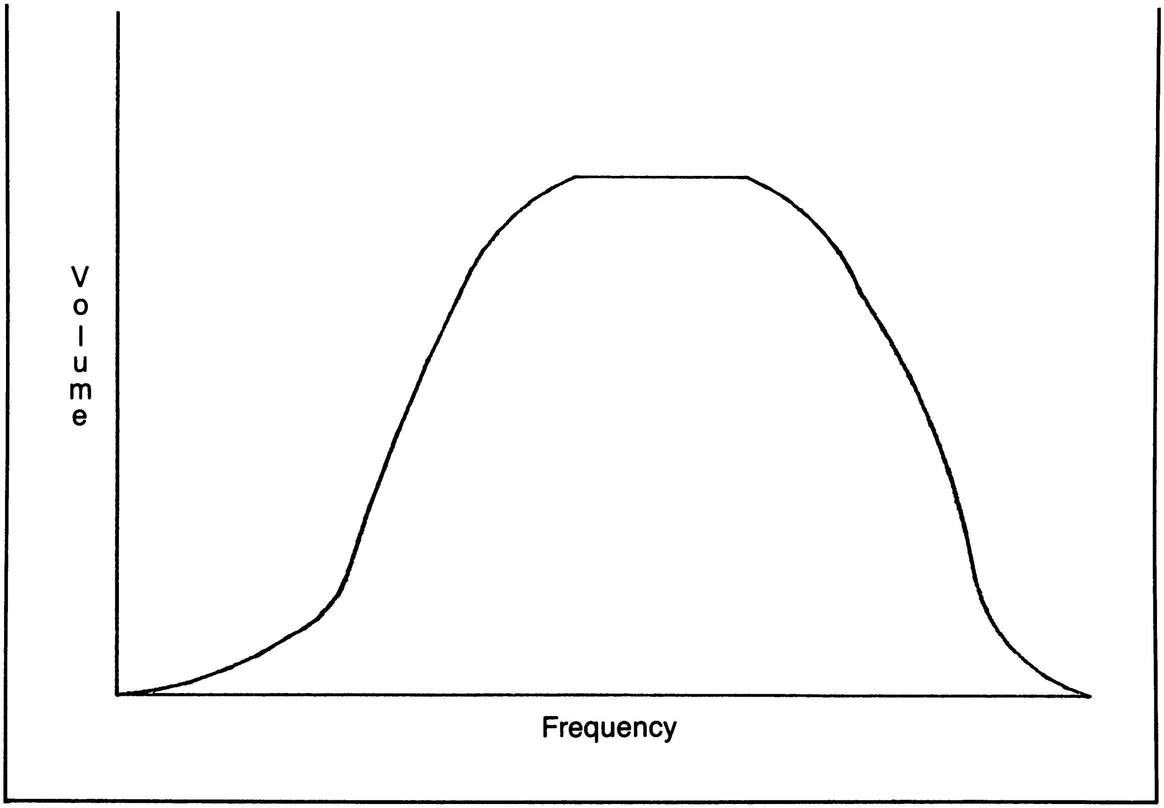


Fig. 8-7. The effects of changing the filter cutoff for a band-pass filter (continued on p. 44).



C-10 is the source coed; Listing C-11 is the assembled code. The Sound Editor shows all the SID chips' registers on the Screen using the naming conventions discussed in Chapter 3. A cursor can be moved around the screen, allowing you to change the values that will be loaded into any of the SID registers. All of the numbers are typed onto the screen in hexadecimal notation, so the data that you create can be easily entered into your assembler. After all of the data fields that you care to change have been updated, you can instruct the Sound Editor to transfer the data to the SID chip. If you have enabled one of the channels, this should produce a sound. To run the Sound Editor, type:

LOAD"SOUND EDIT", 8, 1
SYS 4096

The controls for the Sound Editor are shown in Table 8-1.

Table 8-1. The Controls for the Sound Editor Program.

CURSOR DOWN	Moves the cursor down one field
CURSOR UP	Moves the cursor up one field
CURSOR RIGHT	Moves the cursor one field to the right
CURSOR LEFT	Moves the cursor one field to the left
0-9	Allowable numbers for the data fields
A-F	Allowable letters for the data fields
F1	Transfers the data from the screen to the SID chip and the software timer.

A brief description of the registers used by the Sound Editor follows. For a more detailed description of each register, refer to the section on the SID chip. A listing of hexadecimal values for 6+ octaves

worth of notes is provided in the COMMON file, Listing C-7 in Appendix C. In the following description, wherever all three voices have identical registers, voice #1 is used as an example.

V1ATDC	The high nibble controls the attack time for this channel; the low nibble controls the decay time.
V1SURL	The high nibble controls the sustain level for the channel; the low nibble controls the release time.
V1FRLO	This is the low order 8 bits of a 16 bit value that specifies a frequency for the channel.
V1FRHI	This is the high order 8 bits of a 16 bit value that specifies a frequency for the channel.
V1PWLO	This is the low order 8 bits of a 12 bit value that specifies the pulse width of the channel when you are generating a square wave.
V1PWHI	The lower nibble of this register contains the high order 4 bits of a 12 bit value that specifies a pulse width of the channel when generating a square wave.
V1CORG	This register is the control register for the sound channel. It controls the type of waveform as well as the synchronization mode. The enable bit for the channel is in this register.

The following registers affect all three voice channels.

FLCNLO	The least significant 3 bits of this register are the low order 3 bits of an 11 bit value that controls the filter frequency.
FLCNHI	This register contains the high order 8 bits of an 11 bit value that determines the filter frequency.
MODVOL	The filtering mode and the maximum volume for all three voice channels is controlled by this register.
RESFLT	This register contains the resonance value and the filter enables for all three channels

The last two registers used by the Sound Editor are not SID registers but RAM locations. These registers are treated as a 16 bit value that is used as a time counter. They are decremented every 1/60 of a second. When the 16 bit value has been decremented to $0000, the release sequence is initiated for all three channels.

SNDTM1	This is the low order 8 bits of a 16 bit value that determines the time in 1/60 second intervals before the release sequence is initiated.
SNDTM1+1	This is the high order 8 bits of a 16 bit value that determines the time in 1/60 second intervals before the release sequence is initiated.

By experimenting with different values in the registers, you will very quickly get a feel for what effect different values have on the sound. Being able to specify the amount of time to play the sound can greatly speed up the time it takes to polish a game, as the values for the sounds can be determined separately from the main program.

Chapter 9
Creating Graphics

Up until now,this book has been primarily concerned with background information necessary for creating the program that will ultimately become a game. This is certainly an important part of learning to program a game, but by no means all of it. Because a video game is an audio-visual experience, it will be necessary to create the graphics data that will be operated on by the program.

There are a number of methods that can be used to create and enter graphics information into the computer. The method that you choose is purely a matter of personal preference. This chapter will discuss a few of the options available to you when it is time to create graphics data. Also, instructions for using the graphics utility programs in Appendix C are given in this chapter.

HAND CODING GRAPHICS

At some point, you will probably be entering graphics data into your machine by hand. As this can be a very time consuming process, this section will show you some techniques that may make your job a little easier.

Before you sit down to enter graphics data, you should have a good idea of what you want the final object to look like. You will need to know the graphics mode that you will be working in. Also, you must decide what colors you are going to use. For instance, if you are using multicolored sprites, each sprite can have one color of its own and can use 2 colors that are common to all the other sprites. To ensure that you will have the proper colors available to you, it would be wise to have decided how all of the characters should look before you start.

Once you have chosen a graphics mode to work in, you will be able to start drawing your characters. In Figs. 9-1 through 9-7 you will find some sample graphics layout sheets. These sheets have a fine grid that is proportional to the dimensions of a pixel on the TV screen. The heavy grid is proportional to one character cell on the screen. There are 25 lines of 40 characters each on the Commodore 64.

One thing that you should keep in mind when you are creating graphics for use on the Commodore 64 is that it is often necessary to use more than one

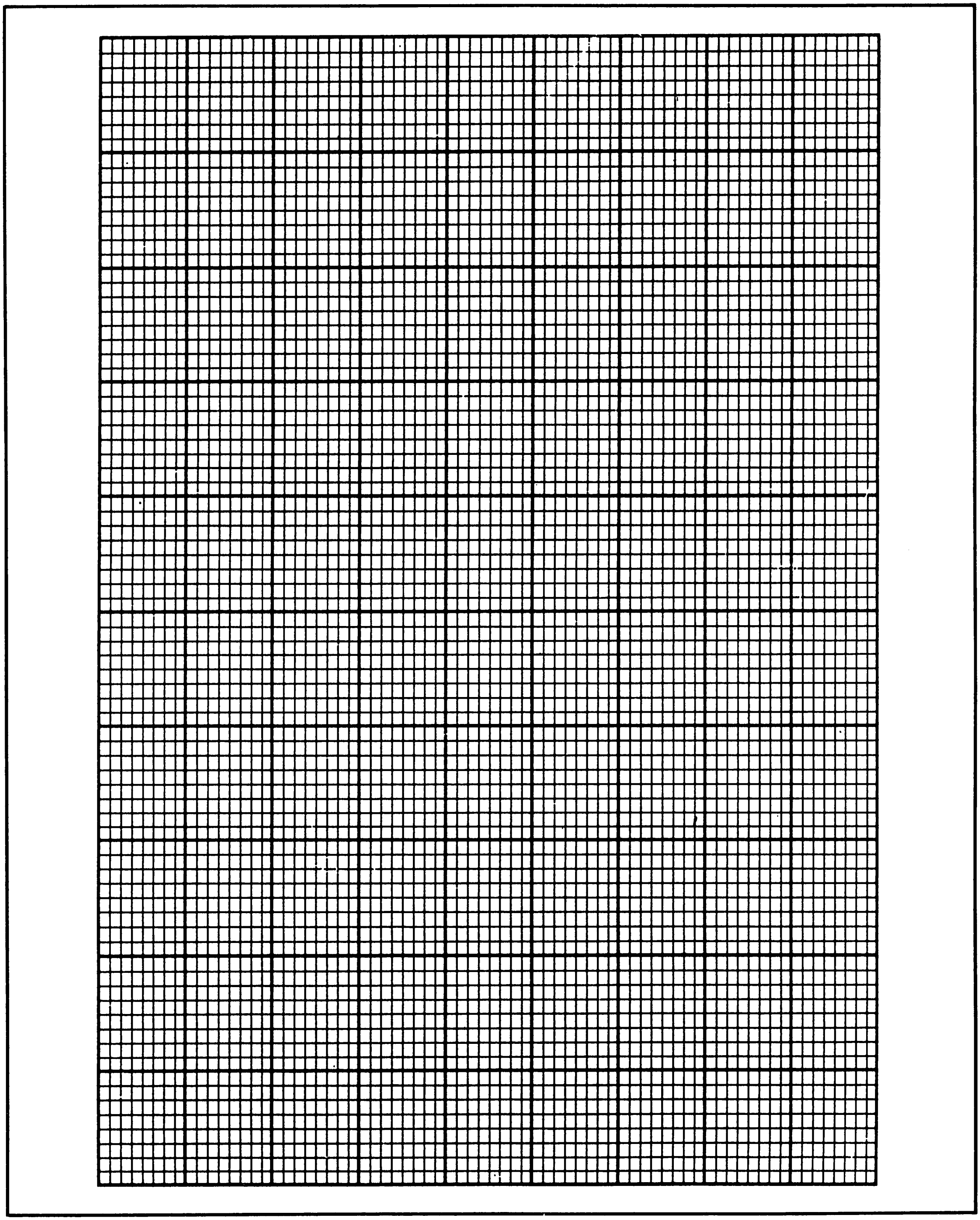


Fig. 9-1. A grid showing character spaces and individual pixels.

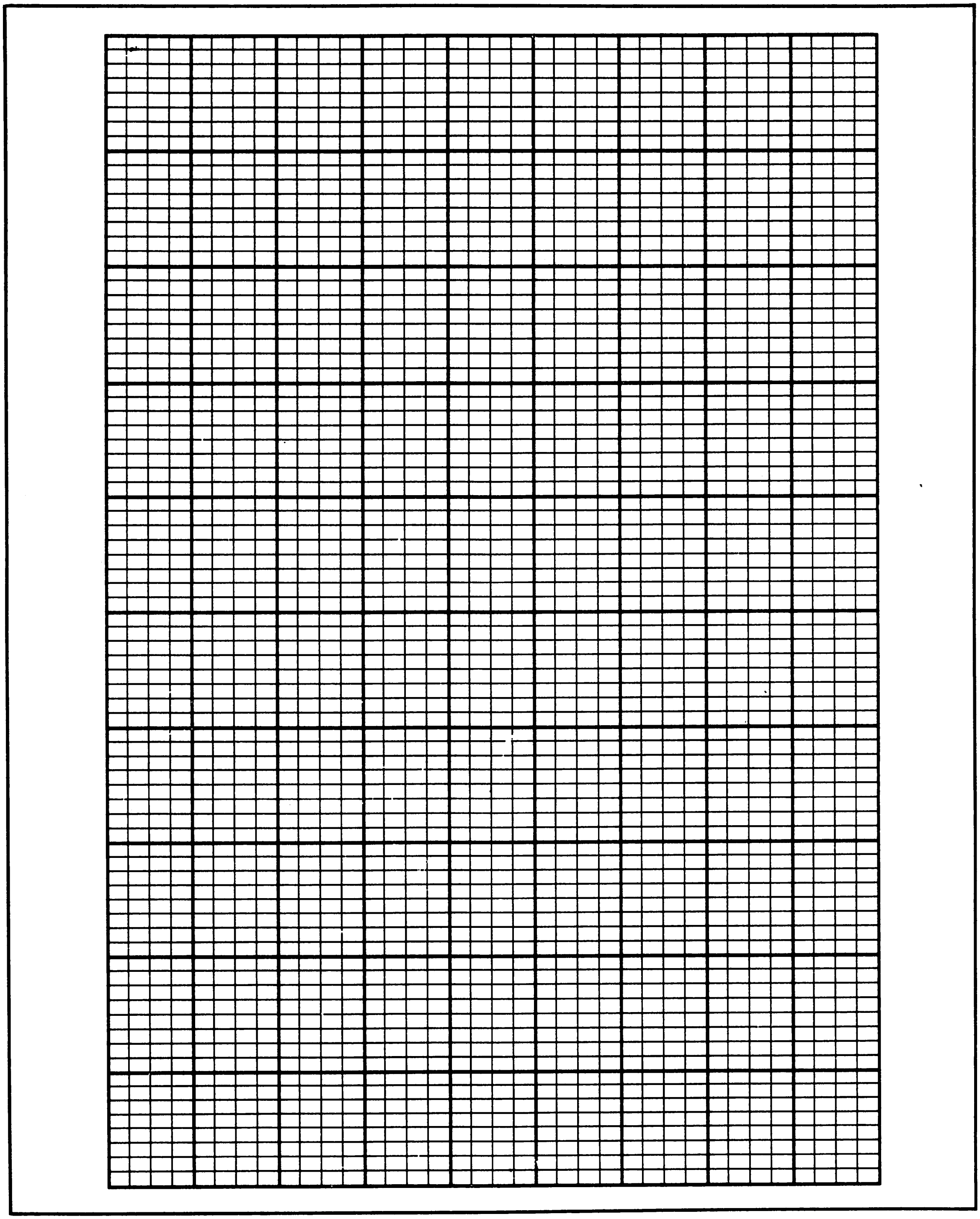


Fig. 9-2. A grid showing character spaces and double-wide dots for multicolor modes.

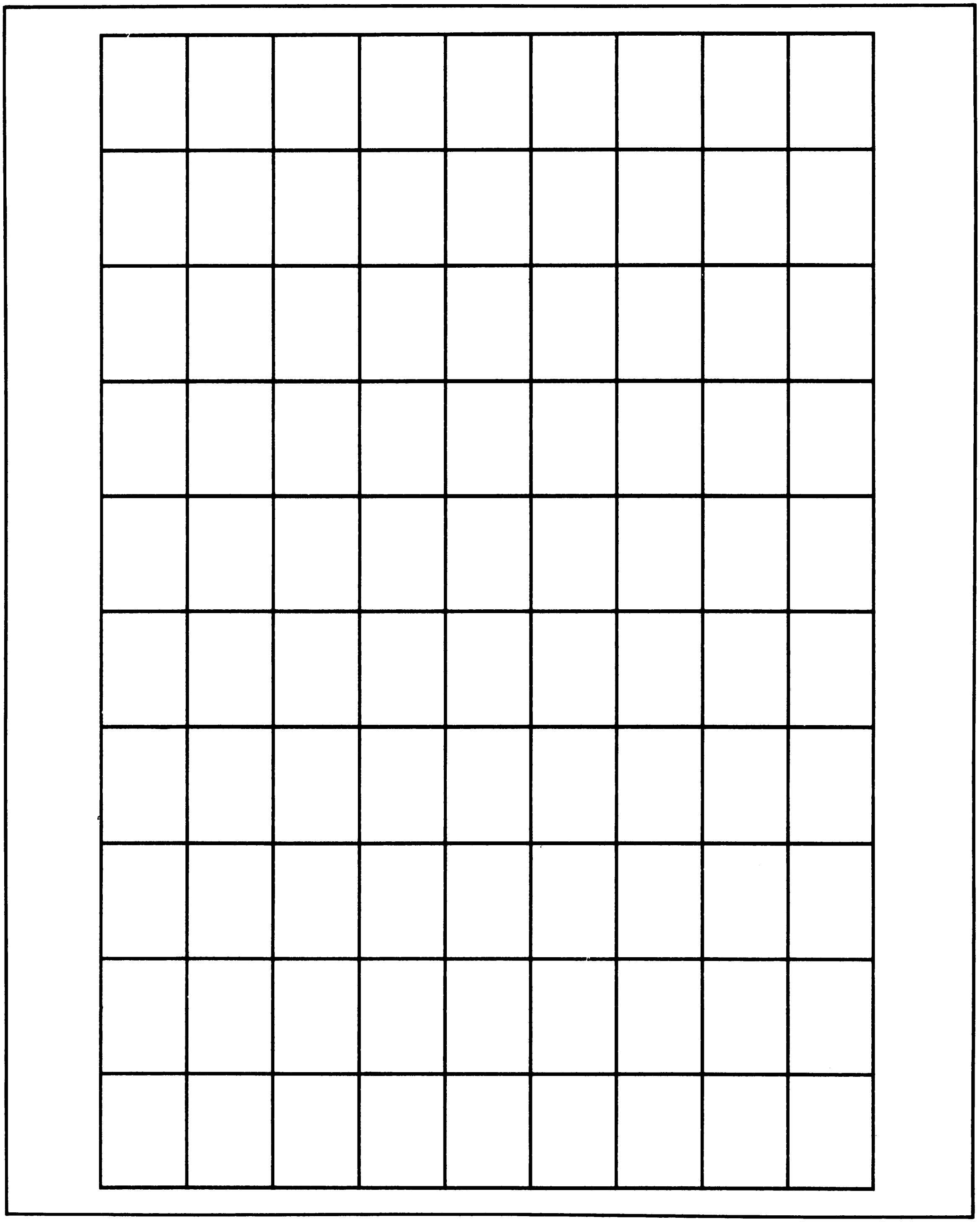


Fig. 9-3. A grid showing character spaces.

pixel of a color on a line to guarantee that the pixel can be seen on the television. Television sets were not designed to be able to display drastic color changes on adjacent pixels. Depending on your color choices, a single pixel in an area of the screen may not be seen. This is due to the time that it takes the TV to turn on and off the appropriate electron guns that brighten a pixel. If the TV does not have enough time to adjust the guns, the pixel will have just started to light when the beams are changed for the next pixel. On the other hand, if you have 2 or more adjacent pixels on a line of the same color, there will be enough time for the beams to be readjusted and the pixels displayed.

The form in Fig. 9-1 is the form to use if you are going to be using the standard bitmapped graphics mode. You will notice that each pixel is rectangular in shape, rather then square as might have been expected. Thus, care must be taken when attempting to draw geometric patterns on the screen. Since the pixels are rectangular, the normal equations for generating geometric shapes do not hold true. However, if you take into account the 4/3 aspect ratio of the pixels, any shape can be drawn properly.

The form in Fig. 9-2 is to be used if you are going to be using one of the multicolor modes for your graphics. You will notice that each pixel on this form is twice as wide as the pixels on the form 9-1. This is because it takes two bits to represent the four colors available to each block in the multicolor mode, as opposed to the one bit that is needed to determine whether the foreground or background color will be used in the standard color mode. Once again, the pixels are not square, and care must be taken when you are creating shapes.

The form in Fig. 9-3 is generally used to represent the screen in the character graphics mode. It also can be used as an overlay to represent one of the color memory areas in the bitmapped graphics mode.

The forms in Fig. 9-4 through 9-7 can be used as design aids when you are creating sprites. There are four different sizes available. The different sizes correspond to the sprite X and Y multiplier options. If you decide to use one of the expanded sprites, be sure to modify your program to change the SPRXSZ and SPRYSZ registers to the size option that you desire. The size of the grid on the sprite form is identical to the grid on the other graphics forms. This allows the sprites that you create to be placed on top of your background graphics forms to see how the entire screen will look.

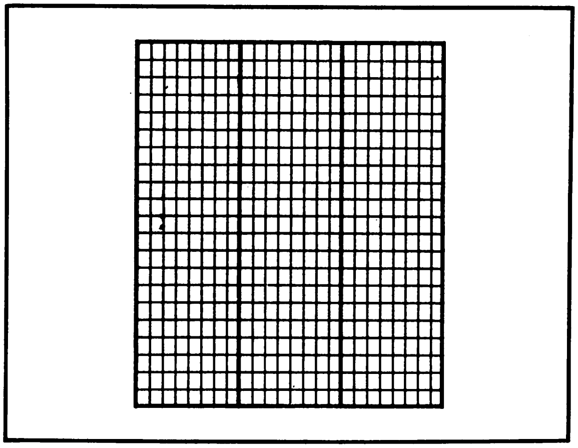


Fig. 9-4. A grid for an unexpanded sprite.

Once you have translated your drawing into a series of bytes, you have a couple of options as to what to do with the data. If you have a machine language monitor, you may choose to use the Memory Display option to display the range of memory where you would like your graphics data to go. At this point, your monitor should allow you to change the data on the screen. By using the data from your drawings to modify the data on the screen, you will create a section of memory that represents your graphics. After you have finished entering the data into the monitor, be sure to save the range of memory that you have just modified to the disk. The next step is to enter the address of your graphics into your program so that it can find the graphics later.

Your other option is to enter the data into your assembler using the .BYTE directive. You can then assign names to all of the different areas of your graphics. The assembler can be directed to store the graphics data anywhere in memory. The main disadvantage of entering the data into the assembler is that the data will be reassembled each time a

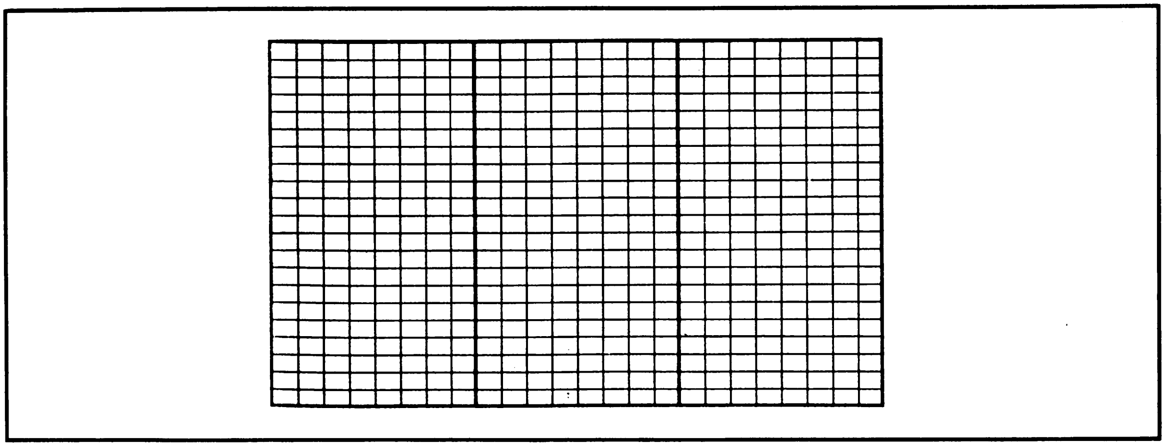


Fig. 9-5. A grid for a horizontally expanded sprite.

change is made in your program. Depending on the amount of graphics data that you have, this can add substantially to the time that it will take to assemble the file. Another option would be to assemble the graphics data separately from the main program. If you do this, you must give the main program the addresses of where the graphics data will be located. In this case, the graphics data need be assembled only once. (Or, at least only as often as is needed to get the data correct.)

USING A GRAPHICS TABLET

A more popular way in which to enter graphics data into the Commodore 64 for use within the program is to use commercially available graphics packages to generate pictures. One of the most popular and useful of these is the Koala Pad. This package comes with a touch pad and software that allows you to easily generate background screens. Almost any type of graphics package will help speed up the process of generating graphics.

A word of caution: before selecting a graphics package to aid with your drawings, it would be wise to be sure that it will fullfill your needs. There are two major pieces of information that you will need about the package:

- What graphics mode does it use?
- How can the picture be retrieved form the disk?

Different types of games will benefit from the use of one graphics mode over another. If your application requires the use of a certain graphics mode, the graphics package that you choose must use the

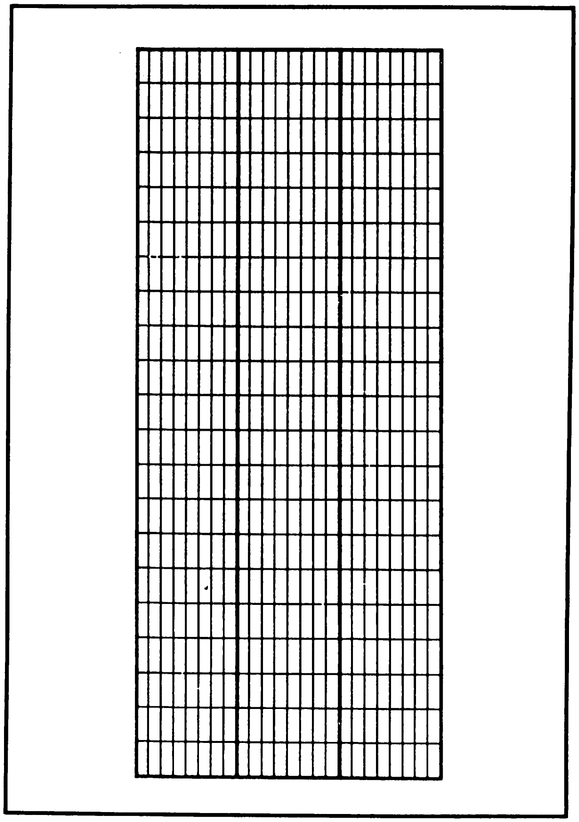


Fig. 9-6. A grid for a vertically expanded sprite.

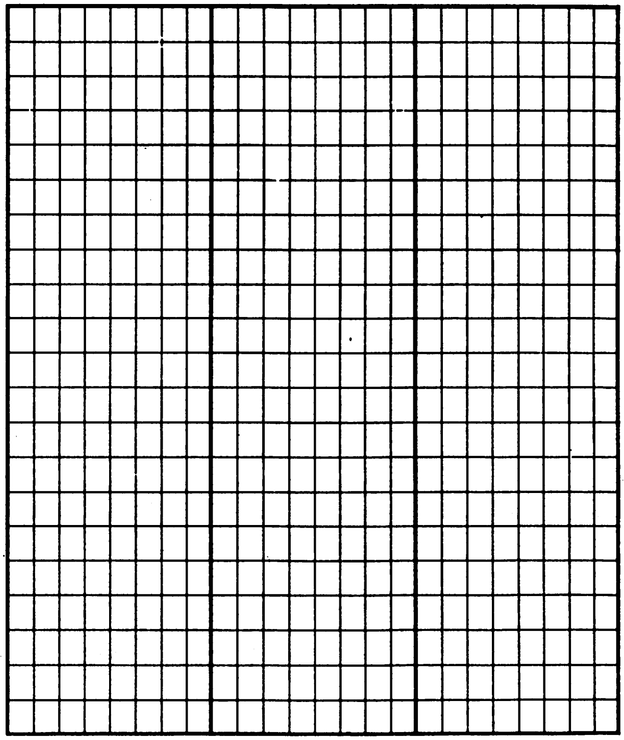


Fig. 9-7. A grid for a sprite that has been expanded both horizontally and vertically.

same mode. Otherwise, the data created will not be useful for your program. This sounds like good advice in theory; however, you will probably find that most of the available packages will use the multicolor bitmapped mode. On the other hand, even if the program can not be used to generate data, it may be useful for testing color choices and to see how things will look.

It is quite possible that you will find a graphics package that will seem to do everything that you would like it to, only to find later on that you can't get your picture back off the disk. It wouldn't seem to be very useful to create a picture if you can't use it. Some graphics packages use what appear to be protected file names to prevent you from retrieving your picture without using the software that created it. You should contact the manufacturer of the graphics package if there is no documentation on how to load the picture without their software. If you can get no satisfactory information from the manufacturer, or they claim that you have no right to know, you should not buy their software package.

When shopping for a graphics package, one especially useful feature is a zoom mode. Using a zoom mode, you will normally be able to change individual pixels on the screen with a minimum of effort. This will allow you, among other things, to clean up a drawing that was made free-hand or create a shape pixel-by-pixel, which is too intricate to create in any other way.

Be wary of a piece of software that will not allow you to use all 16 of the colors that the Commodore 64 can display or does not use the full resolution of the screen. After all, there is no reason to sacrifice any of the abilities of your machine because of some other programmer's shortcomings. In fact, most good graphics programs will give you a pallette of more than 16 colors by giving you mix-

tures of the different colors in different patterns. This can give you a choice of many colors and shades of colors that you may not have been aware were possible.

Another feature that you will learn to appreciate greatly is an OOPS command. This usually allows you to erase the last changes that you have made to your drawing. This is a very useful function when you wish to experiment with color changes and other types of changes or additions that you might not be sure you like. If you don't like your change, you simply give the OOPS command and your drawing is returned to the state it was in before the change.

The following section will give some information about the Koala Pad from Koala Technologies. Some of the information has come from the manufacturer and is not in the documentation that comes with the package. This is not by any means the only software package that will aid in game design; it is only being used as an example. A number of manufacturers have recently released light pens with graphics software, which may be useful. There are also digitizing tablets, joystick controlled graphics software, and keyboard controlled graphics software, which might be suitable. To attempt to evaluate all of the different packages is beyond the scope of this book. The preceding information should aid in your evaluation of a product in the stores, and the following information on the Koala Pad should show you what type of information you will need to properly use the package of your choice.

Using a Koala Pad

The Koala Pad is a touch sensitive tablet with an active area of 4″ by 4″ and a resolution of 256 by 256 points. The pad plugs into one of the joystick ports on the Commodore 64 and is treated as a pair of game paddles by the software.

File format. Before you try to use the data created on the Koala Pad, you will want to convert the name of the file to something more usable. When the software saves a screen to the disk, it precedes your file name with a single byte whose value is \$81. This character prints on the screen as an inverted spade and is inaccessible from the keyboard. Koala uses this character as a flag to identify files that it has saved on the disk. This character is followed by the character string PIC, which is followed by a picture letter and a space.

A pair of utility programs that will convert file names to and from the Koala Pad format are in Appendix C. These are:

Listing C-12	KO-COM	Changes the name from Koala format to Commodore format.
Listing C-13	COM-KO	Changes the name from Commodore format to Koala format.

These are BASIC programs that will prompt you for the current filename and the name that you would like the file to be called. When using COM-KO, the program will insert the special character at the beginning of the name for you.

After you have changed the name of the file into something that can be loaded, you will be able to use your machine language monitor to examine and reconfigure the data. The data is stored on disk in the following format:

Koala Memory Map

\$6000 - \$7F3F	Graphics image
\$7F40 - \$8327	Color memory image
\$8328 - \$870F	Color RAM image
\$8710	Background color

Note: If you are using a cartridge based machine language monitor, you may not be able to read the file as the cartridge replaces the RAM where the data will be loaded.

After you have loaded the file, you can relocate the graphics data and the color data to anywhere that is convenient to your program. You should then save your newly created file back to the disk. You will probably want to move the data since the Koala software is more interested in reducing the size of the disk file than in placing the data in a useful location. It would be a good idea to move the color

memory areas to the beginning of a page boundary. This will make it easier and faster to manipulate the data later. The background color may be stored wherever is convenient.

The data was created using the multicolored bit-mapped mode, so be sure to set the multicolor mode bit in the VIC chip before displaying the picture. Also, note that the border color is not stored in the file. You must set the border color to the appropriate value before displaying the picture.

DISPLAY PIC, Listing C-14 in Appendix C, will display a Koala Pad picture on the screen. It will load the machine language routine MVIT in Listing C-15. This program follows the steps above to display the picture. It will also set the border color to black. It will display the picture until the shift key is pressed on the keyboard. This program will not display a picture if you change the location of the data in the file. To run this program enter:

```
LOAD"DISPLAY PIC",8<RETURN>
RUN
```

Enter the name of the picture to be displayed when prompted. The file name must have been previously changed using the KO-COM utility. A picture in the Koala Pad format is in Listing C-16 in Appendix C under the name PIC A CASTLE. This picture can be viewed using the DISPLAY PIC program. This drawing could make a nice background for a game, if you were so inclined.

USING THE SPRITE MAKER

Listing C-17 in Appendix C is a sprite making utility in BASIC. It loads two machine language routines, SLIB.O in Listing C-18 and CLSP2 in Listing C-19. Using this program, you will be able to quickly create a sprite in either the one color or multicolor mode. You will be able to change any of the colors in the sprite or the background color. After you have finished designing a sprite, you will be able to save it to the disk for use in a program later. As you are drawing your sprite on the screen, you will be plotting squares on a 24 by 21 array of squares on the left side of the screen. In the upper right section of the screen, the sprite is shown in its true size and color, so you will be able to see exactly what the finished sprite will look like. In the upper left corner of the screen, there will be a white box if you are in the plot mode; otherwise you will be in the unplot mode and the corner will be blank.

To run the Sprite Maker program, you must type the following:

```
LOAD"SPRITE MAKER",8
RUN
```

Enter the name to be used when saving or loading from disk.

The SPRITE MAKER uses a combination of joystick and keyboard controls. All of the control options are shown below.

JOYSTICK	Moves the cursor around the zoomed sprite
FIRE BUTTON	Plots or unplots a point
F1	Toggles the plotting mode
F3	Enables the multicolor mode
F4	Disables the multicolor mode
F5	Changes the sprite color
F6	Changes the background color
F7	Changes the multicolor 1 register
F8	Changes the multicolor 0 register
S	Saves the sprite to the disk
L	Loads the sprite from the disk

The joystick will move the cursor around the screen. When the fire button is pressed, a square will either be plotted or unplotted, depending on the plot mode at the time. Pressing the F1 key toggles the plotting mode. The mode is set to OFF when the program is first run. The F3 key will enable the multicolor mode. Pressing the F4 key disables the multicolor mode. The multicolor mode is off when the program is first run. Pressing F5 will increment the sprite color register. The F6 key increments the background color. Pressing F7 will increment the sprite multicolor 1 register. This will only show an effect when the multicolor mode has been selected. Pressing F8 will increment the sprite multicolor 0

register. This will only show an effect when the multicolor mode has been selected. Pressing the S button will save the sprite to the disk using the name that you had entered earlier. Pressing the L button will load a sprite pattern from the disk with the name that you entered earlier.

After you have finished editing a sprite, the sprite data will be at $4000. By using a machine language monitor, you will be able to save the binary sprite data from $4000 to $403F. If you will be using multiple sprites, you may wish to move the data to a safe area in memory so that you can merge your new sprites with the old ones.

USING THE SCREEN MAKER UTILITY

If you are going to use a character graphics mode, you will need some way to specify the placement of the character graphics on the screen. The SCREEN MAKER Listing C-20 in Appendix C will aid you in defining a screen of graphics. This program will load three machine language routines, CLBACK1 in Listing C-21, CLSP1 in Listing C-22, and SLIB.O in Listing C-18. This program will allow you to select any of the characters out of your character set, and place it in any position on the screen. This will allow you to see quickly how your screen will look. When you are finished, you will be able to save the screen to the disk. To run the program, type:

```
LOAD"SCREEN-MAKE",8
RUN
```

Enter the filename to be used when loading or saving a file to the disk when prompted. The following controls are used in this program:

JOYSTICK	Moves the cursor around the screen
FIRE BUTTON	Plots the selected character on the screen
F1	Increments the current character number
F3	Decrements the current character number
F5	Increments the character color
F7	Increments the background color
L	Loads a file from the disk
S	Saves a file to the disk

After you run the program, the screen will clear and the first character in the character set will be displayed in the upper left hand corner of the screen. The next character in the upper left hand corner of the screen shows the current character color. If this color is the same as the background color, the space will appear blank.

Pressing the F1 key will select the next character from the character set. You will see the character in the upper left corner of the screen. By repeatedly pressing the F1 button, you can scan through your entire character set. Button F3 will select the previous character from the character set. By using these two buttons, you will be able to move forward or backward through the character set.

The F5 key will increment the character color. This is the color that will be placed into the color RAM when the character number is placed in the screen RAM. Pressing the F7 key will increment the background color. This is useful when you wish to see what the screen would look like with different background colors.

You may load a screen to be edited using the L command. When you are finished making changes, you should use the S command to save the screen back to the disk. When a screen is saved to the disk, the color information is saved to the disk along with the character placement information.

Chapter 10
Some Arcade Games

Before attempting to design your own video game, it would probably be helpful to understand how some other games are designed. In this chapter, you will be shown some of the ways in which some popular arcade video games could be programmed into the Commodore 64. In fact, the description of how to program these games may be quite accurate in terms of how the original was done. Bear in mind, however, that arcade machines tend to have some specialized hardware for graphics creation.

PAC-MAN

The most popular arcade game in recent times is PAC-MAN. This is a maze type game in which the player controls PAC-MAN in his journey around the maze while being chased by computer controlled ghosts. If PAC-MAN is hit by one of the ghosts, he loses a life. On the other hand, if PAC-MAN manages to eat one of the power pellets on the playfield, for a short amount of time the ghosts will turn blue. During this period of time, PAC-MAN may eat his enemies for a greater score.

Difficulty levels are created by changing the speed of all of the characters and changing the amount of time that PAC-MAN has in which to eat the ghosts. PAC-MAN must eat all of the pellets on the playfield in order to advance to the next difficulty level. Twice during each level of play a bonus character appears on the screen for a short period of time. If PAC-MAN can eat the bonus character, he gets bonus points. The bonus character is worth more points on the higher levels.

If you were to program PAC-MAN on the Commodore 64, you would probably use the multicolor character graphics mode for the maze and the pellets. PAC-MAN and the four ghosts would be sprites. The bonus character could also be a sprite. At this point, you would have two sprites to spare because the Commodore 64 allows eight sprites, and only six have been allocated. These remaining two sprites could be used as the four power pallets by repositioning the sprites after the top 2 pellets have been displayed. This technique will be discussed in more detail in a later chapter.

Since the Commodore 64 maintains collision registers to determine collisions between sprites and between sprites and background, determining when PAC-MAN hits a ghost or power pellet should be no problem. The only part of the program that may be difficult is the part where it is determined which of the normal pellets PAC-MAN has eaten. For the most part, putting PAC-MAN on the Commodore 64 would be a very direct translation. On the arcade game, the video monitor is rotated 90 degrees, which makes the maze taller than it is wide. This would be the only major discrepency between the arcade machine and a Commodore 64 translation.

DONKEY-KONG

In DONKEY-KONG, the player controls Mario, who is trying to rescue a girl from the clutches of DONKEY-KONG, a large ape. To do this, Mario must climb the various structures that DONKEY-KONG sits on. To make Mario's life more difficult, DONKEY-KONG keeps throwing barrels and hammers down at Mario. On some screens, there are fireballs that he must dodge. As Mario climbs toward DONKEY-KONG, he can pick up articles of the girl's clothing that she has dropped on her way up. Mario can also pick up a hammer and proceed to beat up the barrels and fireballs for a short amount of time. Mario can jump over the barrels and fireballs, for which he gains points. If Mario can get up to the level where DONKEY-KONG is standing, he either rescues the girl, or depending on the level, DONKEY-KONG carries the girl higher up on the structure. Mario is then presented with the next level of play. The arcade version of the game has four different types of structures that must be climbed.

Translating DONKEY-KONG to the Commodore 64 is very similar to translating PAC-MAN. The background structures, DONKEY-KONG, and the girl can all be made up of character graphics. The articles of clothing that the girl has dropped can also be made up of character graphics. This leaves all of the sprites free for Mario, the barrels and the fireballs.

If you watch an arcade version of DONKEY-KONG very carefully, you may see how its designers avoided the problem of needing a large number of sprites. When a barrel rolls past Mario and heads for the next lower level, it will normally roll off the edge of the screen rather than descending to the next level. Because the player's eyes are normally focused on Mario and the portions of the screen above him, he will not normally notice the disappearing barrels. Because the machine does this, it never needs more than five sprites to display all of the barrels. This same technique will work for a translation of DONKEY-KONG for the Commodore 64. If five sprites were reserved for fireballs and barrels and one for Mario, there would still be two left over for hammers. In fact, you would have even more flexibility in the use of sprites. For the most part there are only 3 or 4 sprites displayed on any given line. Using the technique of repositioning sprites, you could reposition the sprites on different lines.

Like PAC-MAN, in the arcade, DONKEY-KONG's screen is rotated 90 degrees from a normal television. For this reason, any translation of DONKEY-KONG to the Commodore 64 will be wider and shorter than the original.

CENTIPEDE

In CENTIPEDE, a nasty centipede is running loose in a field of mushrooms. It starts at the top of the screen and winds its way down the screen until it gets to the bottom, where the player's gun is. The players must shoot the centipede without getting hit by it. The centipede has 11 body segments. If the head is hit, the next body segment becomes the new head. If a body segment is hit, the centipede breaks into two parts, each of which has its own head. Whenever a segment of the centipede hits a mushroom, it drops down to the next line and turns around.

In addition to the centipede, the player must also avoid the spiders and fleas. Fleas add to the mushroom field, while spiders destroy mushrooms. Scorpions poison the mushrooms that they touch. A poisoned mushroom causes the centipede to descend straight down the screen.

This game could be a bit of a problem to translate because of the large number of moving objects. The mushrooms can be made using multicolor programmable characters. Since the Commodore 64 has only eight sprites to work with, there would appear to be a shortage of sprites to use in the translation. However, by using the technique of sprite multiplexing, you can trick the computer into working as if it had 16 sprites. This technique is discussed further in a later chapter; if used properly, it would allow the game to be translated for the Commodore 64. Eleven sprites are used for the centipede's segments, 1 sprite for the player, 1 sprite for the shot, 1 for the flea, 1 for the spider, and 1 for the scorpion. This totals 16 sprites, or the number of sprites that multiplexing would provide.

THE REVENGE OF THE PHOENIX

To help illustrate how some of the different techniques describe in this book translate into a game, an arcade style game, Revenge of the Phoenix, has been included in Listing C-23 in Appendix C. This game uses almost all of the techniques that have been covered earlier. It is included to help illustrate the capabilities of the Commodore 64. To play the game, type the following:

```
LOAD "PHOENIX V1.4N", 8,1
SYS 32768
```

At this point, the program will go into its introduction mode. If left alone, it will demonstrate how the game plays and eventually return to the introduction. You may interrupt this process at any time by pressing one of the fire buttons. The game can be played by either one or two players. You can choose which mode you want to play by moving the arrow with the joystick on the title page. By pointing at the character that represents the mode that you would like to play and pressing the fire button, you will initiate game play in that mode. Since the two players (high wizard and low wizard) do not have exactly the same capabilities, you may choose which of the two players you would like to control.

Game Play

In Return of the Phoenix, you will be playing the character of a wizard protecting a castle from the magical phoenix. You are able to stun the phoenix with spells shot from your staff. The goal is to prevent the phoenix from building a bridge in the sky. They will try to get to the bottom of the screen, pick up an energy spell, and bring it to the top of the screen, where a section will be added to the bridge. Game play will continue until the three tier bridge is completed. At this point, both you and the phoenix will have a score. If your score is higher than theirs, you win.

One of the wizards can fly around the screen and go virtually anywhere that a phoenix can. The other wizard must stay near the bottom of the screen. This gives each of the wizards unique playing characteristics. Which one that you care to play is very much a matter of personal preference.

There are nine levels of difficulty in Return of the Phoenix. The skill level that you are currently playing is shown at the top of the screen in the center. As you are playing, the program constantly monitors your playing ability and modifies the skill level accordingly. If the program feels that you are playing very well, it will increase the level of difficulty. On the other hand, if you are playing poorly, the skill level will be decreased. The skill level can only be decreased if you are above level 4. Starting at level 2, the phoenix will start dropping sleep spells on the wizards. If you are hit, you will be unable to move for about three seconds. At the higher levels, the phoenix will shoot more often and move faster and at some levels, the wizards will move faster also.

Scoring

In this game, you are competing against the phoenix for the high score. The phoenix are controlled by the computer, and they have a different method of scoring then do the wizards. The number of points that the phoenix gets depends on how long you can prevent them from getting energy bricks to the bridge. After every four seconds, the value of the bricks to the phoenix decreases. The bricks can be worth from 5 to 98 points depending on how long you can keep the phoenix from getting a brick to the bridge. All of the rest of the scoring is more

standard and shown in Table 10-1.

When the two player mode is selected, both players are working for one score. The players' score is in the upper left corner of the screen. Unlike most video games, in which the two players are competing, both players are working together for a common goal. The phoenix score is in the upper right corner of the screen. This is just as valid a score as the players' score, in that they are working toward their own objectives. If the wizards beat the phoenix score, the bridge will flash and fall down at the end of the game. Similarly, if the phoenix beat the wizards score, they will display their message at the end of the game.

Table 10-1. The Scoring System for the Revenge of the Phoenix Game.

Wizard shooting phoenix with energy brick	98 points
Wizard shooting phoenix without brick	26 points
Wizard shooting all 4 phoenix	1000 points
Phoenix putting a wizard to sleep	325 points

Chapter 11

Elements of Game Design

Much of this book has been dedicated to the techniques of programming a video game on the Commodore 64. In this chapter, some of the concepts that need to be used during the design of the game will be discussed.

Be fun
Have an interesting plot
Be visually stimulating
Have sound effects and music
Have varying difficulty levels
Keep score

When you are designing a game, you are writing a program that is intended to be used for the amusement of others. Trying to make the game fun to play should be your primary consideration. There are some problems in trying to design an enjoyable game, however. For instance, when you come up with a game concept, you may think your game will be the best game ever written, only to find, after you have written the program, that it is boring. This can be caused by a number of factors. More often than not, the game idea was good, but the computer lacked the ability to display a game as complex as you wanted.

One thing to beware of when designing a game, is the tendency to have the game play in exactly the same way each time it is played. If there are no random elements in the game, each move by the player will cause a specific move by the computer. This may be challenging at first, but will quickly become boring once it is mastered. The PAC-MAN arcade machine had this problem and it didn't take long before patterns that showed how to beat the machine every time were published. All it takes is an occasional random move for the play to be unpredictable—which will add to the challenge of the game.

After you have spent some time programming the Commodore 64, you will have a good idea of what it is capable of doing. If you take into account the capabilities of the computer during the design phase of your program, you will have a much easier

time writing the program. Many of the routines explained in this book will enable you to write more complex games than you may have thought possible. Techniques such as sprite multiplexing make it easy to display more sprites in the same area of the screen than is otherwise possible. This will give you more flexibility in the design of your game, which will make it more fun.

Your game can usually be made more interesting if you discuss some of your ideas with others before you start programming. Because everybody sees things differently, you may be given some ideas that will greatly enhance the play of the game. Many of the large game corporations put a number of game designers in a room and have them toss ideas back and forth. A bull session such as this can be the fastest way to get creative input into the design of a video game.

VISUAL IMPACT

The visual impact of the game is the first thing anyone playing your game will notice. If the animation is interesting, it will quickly attract attention. The proper use of color is important. When you are using a normal television as a monitor, certain colors interact with each other better than others. Black characters on a white background will give quite a bit of contrast. The characters will be very clear and sharp. On the other hand, red characters on a blue background will appear fuzzy and indistinct.

Varying the animation sequences that are used to make up a moving figure can add to the attraction of the game, if they are changed at the appropriate time. A game in which a character explodes and fades out after it is shot will be a more interesting game than one in which the character simply disappears. Similarly, figures can be changed to indicate that something is being carried, or that the player is at a different level. At times, simply flashing the colors of a character will add to the visual effect of the game.

SOUND EFFECTS

Sound effects and music can add greatly to the appeal of a game. If nothing else, music played while the title page is displayed will hold a player's attention as they are reading the credits, rules, or whatever else you may choose to put on a title page. In some cases, background music may be appropriate to a game. When this is done, it should be played at a relatively low volume with respect to the rest of the sound effects. Loud background music can be quite a distraction and a nuisance when it drowns out the sound effects. You may want to consider creating an option that will allow the game to be played without any background music.

Many games provide a brief break as the player moves from one level of play to another. Quite often, during this pause, a short piece of animation is shown on the screen along with some music. If a game takes a long time to play, these breaks give the player a time to relax and catch his breath.

DIFFICULTY LEVELS

Virtually every game has different difficulty levels. The variations can range from simply increasing the speed of some of the characters at different scores to starting a completely different portion of the game after a task is completed. For the most part, a game will be the same each time that it is played. Changing the level of difficulty based on the skill of the player will keep the player interested in playing the game even after he has mastered the beginning levels. Presenting him with a new challenge as a reward for gaining skill in the game will help to keep the player interested. You can make the game more fun to play by increasing the speed of the characters at higher levels and increasing the amount of shooting that the player is allowed to do. Some games will introduce a new character at each new difficulty level, so the player will keep playing in order to see all of the different characters.

In the game Return of the Phoenix, Listing C-23 in Appendix C, each new level of difficulty has a new speed for the characters and a different rate of fire. The level of difficulty is decided by the players' skill. If he is playing well, the level of play will increase. When the player starts doing poorly,

the level of play will be reduced. This self adjusting difficulty (SAD) system keeps the game interesting by constantly adjusting the level of game play to the player.

SCORING

A player can tell how well he is doing by looking at the score. Almost all games should have a score of one sort or another. A player should be awarded points for actions that help him toward the goal of the game. The number of points awarded can vary greatly for different types of actions and can even be based on the current level of difficulty. In most cases, you will want to display the score constantly so that the player can tell how he is doing.

Usually, when an action results in points being added to the score, some appropriate sound effect is generated. This sound serves to inform the player that he has done something good without forcing him to look at the score. Under some circumstances, a bonus should be awarded for completing a specific task. Bonuses are often used to tempt a player into a course of action, for some immediate points, which is not necessarily beneficial in the long run.

Deciding on the number of points to award for the various actions is very subjective. You want a normal score to be high enough to make the player feel that he has accomplished something, but not so high as to be incomprehensible. You will probably not decide on your final scoring method on your first try, but will refine it during the testing process.

Note: When you first decide on a scoring method, be sure to reserve enough bytes for the score to accommodate an extremely high scoring player. This will avoid the rollover when the score changes from all 9s to all 0s. Never assume that just because you can't get over a certain score that no one can.

As an added feature, you may wish to include a feature that allows the highest scoring players to place their names on a scoreboard. Although not necessary, this feature can add to the competition between a number of players.

The next chapter describes in detail the operation of the BOGHOP game, which is in Appendix C. This game has been designed to demonstrate virtually all the programming techniques described in this book.

If you have never programmed in assembly language, you may wish to try assembling the program as an exercise. Before doing so, you should make a copy of the source code disk and only work with the copy. The program has been structured in such a way that small changes can completely change game play. By reading the comments in the program listing, you will be able to see where you can make changes. This provides you with an easy way to start experimenting with an assembly language program.

This concludes the introduction to arcade game programming on the Commodore 64. Just as in any other field, the best way to learn is by doing. In this book, you have been given a strong foundation on which to build your program. By using the various definition files and libraries from Appendix C, you can spend more time writing your game program and less time coding the groundwork. I hope that you find a great deal of enjoyment in writing games and sharing them with others, as we have.

Chapter 12
How BOGHOP Works

This chapter describes the routines in the BOGHOP game in Appendix C. A number of the routines are general purpose in nature and you may find applications for them in other programs. For each routine, the purpose of the routine will be listed; then a description of how the routine was implemented will be presented. Assembler directives will also be described where applicable. Before reading this section, you should play the game for a while. This will help you to visualize what the program is doing. BOGHOP in Listing C-24 is the source code for the main program. BOGHOP.O in Listing C-25 is the assembled version. The other files used are described below. To run the game, type:

```
LOAD"BOGHOP.O" , 8,1
SYS5120
```

The number of points that you get for shooting any given bad guy increases with each successive wave.

The code for the game in Appendix C is spread over a number of files. In addition to BOGHOP, these files are:

Listing C-1	MACLIB
Listing C-2	SYSDEF
Listing C-26	BOGDEF
Listing C-27	BOGDAT
Listing C-7	COMMON
Listing C-28	XXPLOT
Listing C-29	LOOKUP
Listing C-30	BOGSPR

MACLIB is the macro library. The source code for all of the macros described in this book can be found in this file. Detailed descriptions of the macros can be found in Appendix B. All of the assembly language programs in Appendix C use this file.

SYSDEF contains the system definitions. Names are assigned to all the hardware registers that will be used throughout the program. This file is also used by the other programs in the book.

BOGDEF has all of the RAM definitions (the variables and buffer areas) that are used in the BOGHOP program. The definitions in this file are specific to the BOGHOP program and will not be used by any other program.

BOGDAT is a date definition file. Lookup charts and other data to be used by the BOGHOP program are defined in this file. Like BOGDEF, this file is used exclusively by the BOGHOP program.

As its name implies, COMMON is a file that is used by virtually all of the assembly language programs in this book. In this file, names are assigned to notes over an eight octave range. This is a useful file for any program that uses the SID chip.

XXPLOT contains a subroutine named XPLOT, which is used to exclusive OR a point to the screen. It uses data from the LOOKUP file, which must be loaded into memory before using XPLOT. The advantage of exclusive ORing a point to the screen is that if the same point is plotted twice, the screen will be returned to the same condition as before the first point was plotted. Note that OOPLOT, Listing C-31, allows points to be ORed rather than exclusive ORed to the screen. BOGSPR contains the data that defines the shapes of the sprites that will be used in the BOGHOP program.

The code portion of the program is in the BOGHOP file (Listing C-24). This file loads the other routines during assembly. The following description tells how the BOGHOP file loads the other files and how they relate to the BOGHOP program, and how the BOGHOP program runs.

THE START OF THE PROGRAM

The first line of the program is a PUT statement with the name of the program. After you have finished editing the program, you can delete the line number and semicolon, (which turns the line into a comment) and press the RETURN key to save the program to disk. By saving the program in this way, you will always save the program to the disk with the same name. Because it is common to use similar names for different sections of a long program, the possibility of saving the program under an already used filename and thereby destroying the other file is eliminated.

Following the put statement are a number of other comment lines used to document the program. It is a good idea to put the date of the last update to the program in the program itself, so you will be able to distinguish one printout from another. If you are making a number of different changes to the program in the same day, you should also put the time of the last update in the program. The name of the program module should be in the first few lines of the program so that you can quickly tell what program you are working on.

The Macro Library

The .LIB directive is used to load in libraries of code, data, and definitions. In this program, a library of macro definitions, a library of system definitions, and a program segment of RAM definitions specific to this game are loaded in. A library of note definitions and note time values as well as data specific to the program are also loaded.

When a .LIB directive is used, the assembler will treat the code to be inserted as if it were typed into the program at that point. By inserting other program segments into your program in this way, you will not have to make any changes to your libraries to use them in other programs. You also gain an advantage in the speed of editing. Had you entered all of the different sections as one large program, every time that you tried to edit the program, you would have to wait for the entire thing to be loaded into memory. In the Commodore 64, this could take quite a while. Also, it is easier to print listings of only the portion of the program that you are currently working on, if the program is broken into smaller segments.

The macros in the macro library and the system definitions have been described in detail elsewhere, so they will not be described here. The BOGDEF file contains the zero page RAM allocations that will be used for the game. Whenever possible, the zero page of RAM should be used for all variables. Doing so will cause the code generated by the assembler to be 2/3 as large and execute significantly faster than if the variables are elsewhere in memory.

In this program, variables are allocated using the DS macro rather than using equates. The DS macro reserves the number of bytes following the macro call for the label preceding the macro call.

This frees you from having to keep track of where each variable is going to be placed in memory. More importantly, if you find at a later time that you need to insert a variable between two existing variables, you will not need to redo all of the equates following the new variable. You would need to redo all the equates if you were using indexed addressing and a new variable had to be in a specific place. When you are using equates, it is easy to make a mistake and define two or more variables with the same memory location. A mistake of this type can be very difficult to debug. When using the DS macro, the assembler will keep track of the addresses of the variables and generate them each time the program is assembled.

RAM Definitions

In the beginning of the RAM definition section of the program (the BOGDEF section) program, all of the shadow registers for the SID chip are defined. As mentioned earlier, you will usually change a value of one of the shadow registers for the SID chip rather then changing the SID chip directly. Most of the registers in the SID chip are write only registers. For this reason, you will not be able to find out what the current value is in one of the registers. In this program, the value in the shadow registers will be transferred to the appropriate register in the SID chip during one of the interrupt routines. The technique of shadowing registers will always leave the current value of a hardware register in a variable location, so that the software will always have access to the value.

Also defined in the RAM definition section of the program are a number of general purpose registers. These locations can be defined in the same way for virtually every program you write. You may not use all of them every time, but they provide a good starting place.

All of the two byte definitions will be used as pointers for indirect addresses. SRC and DST are used as source and destination pointers for data transfers. The NOTPT and NOTTM registers are used to hold the address of a sequence of notes and their respective durations for generating music.

The SNDTM registers are used as software timers to control the voice channels. Whenever a value is placed in one of the SNDTM registers, it is decremented until it reaches 0. Once the value reaches 0, the appropriate voice channel will be disabled, causing the release sequence to be initiated.

The OPTION register can be used to hold the number of whatever option is being used during the game. Options can be things such as one or two players and easy or hard play.

The RAND registers contain random numbers. During one of the interrupts, a new value is placed into one of the four RAND registers. Every four screens, there will be new values in each of the RAND registers. The value placed in the RAND registers is derived from the RANDOM registers, the SCREEN, timer, and the other RAND registers. This helps ensure that there will be a fresh value in the RAND registers whenever you need a random number. The RANDOM register, which is part of the SID chip, changes at a rate proportional to the frequency of voice 3. If this register were to be used directly to get a random number, there is a good chance that your program would be so fast that the value could not change since the last time it was read.

In the LEVEL register, you will usually find the current difficulty of play. The next four registers are used as system timing registers. SCREEN is incremented once every 1/60 of a second. RANSEC is incremented every 1/60 of a second, but counts from 0 to 59. SECOND is incremented once per second when RANSEC goes from 59 to 0. ENABLE is set to a 1 at the end of the INTO interrupt routine. This register is used to synchronize the main program with the interrupt routines, thereby keeping the timing of the system absolute. At the end of the main program loop, ENABLE will be set to 0. The program will then enter a short loop until ENABLE goes to 1. This will ensure that the main program loop is executed only once per screen (1/60 second).

There are four buffer areas defined in RAM. BUF and BUF1 are to be used as temporary storage and work areas for any of the routines in the main program loop. Each buffer area is defined to be eight

bytes long. MBUF is a buffer area that is to be used only during macro calls. Similarly, IBUF is to be used only during interrupt routines.

Some restrictions must be observed for the proper use of the buffer areas. These buffer areas are for the internal use of routines only! They should not be used to pass data from one routine to another. If data needs to be passed, a variable should be defined in which to pass the data. The proper buffer area must be used for each routine. Allowing an interrupt routine to use BUF or BUF1 can result in unpredictable operation of your program. If an interrupt is received while a routine that uses BUF or BUF1 is in process, the interrupt can destroy the data if the interrupt routine also uses these buffer areas. If you find that you need a larger buffer area, you should probably define a special area for the routine that needs it.

The LPCNT registers are for use as loop counters. Under normal circumstances, you will be using the X or Y registers as loop counters. In some cases, you will be changing the X and Y registers during a loop, making them useless as loop counters. As with the buffer areas, do not use the LPCNT registers in both the main program loop and in an interrupt routine.

The next series of variables are used to specify a point to the point plotting routine. COLOR requires a number from 0 to 3 to determine the bit pair to be plotted on the screen. POINT and CTEMP are internal registers to the point plotting routine. GBASE is a two byte address that specifies the base address of the high resolution screen to be used by the point plotting routine. If you are using a general purpose plotting routine, XOR can be used to specify whether the point should be OR'd or EXCLUSIVE OR'd onto the screen. XPNT has the horizontal position of the point and YPNT has the vertical position of the point. In this game, the point plotting routine is written to subtract the offset of a sprite position, so that the coordinates of the point can be transferred directly from the sprite position to the point position.

HMSB is a register that holds the ninth bit of a nine bit horizontal sprite position. This is used as a temporary register for calculating the ninth bit. The result is usually transferred into another register. Most of the sprite horizontal positions are stored after being divided in half. This allows the horizontal positions to be stored in 8 bits. Before being stored in the VIC chip, the data must be expanded back into a nine bit format. One of the main advantages of storing the data in an 8 bit format is that a horizontal position can be checked using a single byte comparison; the position does not have to be expanded into two bytes before a comparison. There are macros that unpack a nine bit value into either a two byte value or a one byte value after dividing the 9 bit value by 2, but they are less efficient then storing the divided value initially.

HORNC and VERNC hold the increment to be used when changing the position of one of the computer controlled characters. By changing these values by varying amounts depending on the level of play, the characters are given different speeds in the horizontal and vertical directions.

The following definitions are more specifically related to this game, although many of them will be useful in other types of games.

The number of lives that the player has left can be found in LIVES. When LIVES=0, the games is over.

Four bytes are reserved to hold the SCORE. Since the data stored in SCORE is stored in a BCD format, the score can have up to eight digits. Many games preset the least significant digit to 0, which would give nine digits. In this game, the score is run through a leading zero suppression routine that prevents insignificant zeros from preceding the score.

MOUNTV has the vertical position of the mountains, while MOUNTH has the horizontal positions. MOUNTP contains the sprite pointers for the mountains. Colors for the mountains are held in MOUNTC. Each of these registers are defined to be eight bytes long, one byte for each of the mountain sprites. Since the horizontal positions of the mountain sprites are stored in a nine bit format, the MNTMSB holds the ninth bit for the horizontal position.

WHOLIV is a byte that has the status of each character on the screen. If a character is still alive,

its corresponding bit will be set in WHOLIV. The player's characters is in bit 0,the computer controlled characters (BAD GUYS or MEANIES) are represented by bits 1-7. This scheme of using one bit per character is also used by SHOTS to tell which shot is in flight.

TMSCOL is a temporary register used to hold the last valid sprite to sprite collision data as read from SSCOL. Since the data in SSCOL is cleared by reading it, it is necessary to put the data in a safe place until it can be used. This is even more critical when using the sprite multiplexing and repositioning techniques, as the data in SSCOL may not be relevent to your program if you wait until you are ready to use the data before reading it. Similarly, TMBCOL is used to store the data read from SBCOL for sprite to background collisions. SSCOL and SBCOL are transferred to TMSCOL and TMBCOL during INTO.

There are eight bytes reserved for each of the following player and bad guy parameters: horizontal positions, vertical positions, colors, directions, and explosion status.

Sixteen bytes are reserved for use as pointers to the movement charts used by the bad guys to determine their movements.

SHOTSH holds the horizontal position of the shots, stored in an eight bit format. SHOTSV holds the vertical position of the sprites. SHOTSD contains the direction that each shot is to go. Eight bits have been allocated for each of the last three parameters.

This concludes the RAM definitions for all of the variables used by the program. Usually, it is a good idea to try defining all of the variables before you begin to write any code. By doing so, you will force yourself to decide on an approach to take in programming your game before you start coding it. You will find that programming will go faster once you have defined your variables. Of course, if you miss a few, you can add them later.

Musical Definitions

The COMMON file is loaded in next. This file contains frequency definitions for the SID chip. These definitions cover an eight octave range. The format is as follows:

1. The first letter is the name of the note.
2. The second letter is either N for NATURAL or S for SHARP.
3. The third position is for a number between 0 and 7 which specifies the octave number.

COMMON also contains definitions of note times. These times are based around a timer of 1/60 of a second. That is, if you decrement these values once every 1/60 of a second, you will get the proper duration of a note. If you are going to be creating complex musical rhythms, you should verify that the times used for notes in all channels is such that the beat is kept constant. You may have to adjust the times to match your application.

The Data Section

BOGDAT, the next file loaded into the program, is a data section. In it are all of the data and lookup charts that will be used throughout the program.

BITPOS is a lookup table that can be used to set a bit. BITAND is a lookup table that can be used to clear a bit.

Following these lines are a number of lookup charts that are used to determine that bad guys' motion. Each chart ends with a 0 so that the routine that uses these charts can find the end.

The next charts are used to determine which movement charts are to be associated with which type of bad guy.

BASLK is a chart that contains the base addresses of the lookup charts that contain the base addresses of the movement charts.

BADSEQ is a chart that is used to determine which type of bad guy will appear at each level. By extending this chart and the next three charts that will be described, the game can be made to have any number of levels (up to 255).

SPEEDH is a lookup chart that is used to determine the horizontal speed of the bad guys depending on the level. Similarly, SPEEDV is used to choose a vertical speed for the bad guys.

FIRPOW is used to determine how often the bad guys should shoot. A random number is chosen,

and if the number is less than the value in FIRPOW, the bad guy can shoot.

NUMBER is a lookup chart that indicates where the image of the various numbers can be found in memory. NUL is all zeros, for use by the leading zero suppression routine.

TYPELK is a chart of sprite pointers that point to the first sprite in an animation sequence for the bad guys.

SHTDR is a table used to choose a direction for a bad guys' shot.

COLK and POR are used by the point plotting routine.

The Point Plotting Routine

XXPLOT is the next file to be loaded into the program. It contains only one routine, the point plotting routine XPLOT. XPLOT refers to lookup tables that can be found in the LOOKUP file. The lookup tables contain data that serves to map the screen into normal cartesian coordinates. By storing the horizontal coordinate the XPNT and the vertical in YPNT, the proper point on the screen can be found. The base address of the screen must be stored in GBASE. Since the routine is written to make use of a multicolor bit-mapped screen, valid horizontal values will normally be within the range 0-$A0. XPLOT has been rewritten to subtract the offset associated with sprite positions before processing the point.

At the beginning of the routine, the sprite adjustment is made, and the resulting point is checked to see if it is in a range to be plotted on the screen. If the point can not be plotted on the visible screen, the routine will end with an RTS instruction. The vertical position of the point is transferred to the X register. The horizontal position is divided by four and transferred to the Y register. The X register is then used as an index into VLKUPL, which has the lower eight bits of the base addresses of each scan line. This byte is added to the low byte of GBASE, which contains the base address of the screen. The result is stored in POINT. This sequence is repeated for the high byte of the scan line base address, which is in VLKUPH. The result of the addition of a byte from VLKUPH and GBASE+1 is stored in POINT+1. The value in the Y register is used as an index into a table of horizontal offsets of the byte on the screen. As with the vertical lookups, the appropriate offsets are added into the address in POINT. An indirect address that points to the byte that contains the point to be plotted in now in POINT. The horizontal position in XPNT is loaded into the accumulator and AND'd with a $03. The result is then transferred into the X register for use as an index into a table that will specify the bits to change.

In preparation for loading a byte indirectly, the Y register is cleared to 0. An indirect load is then performed using the value in POINT, the loaded value being stored in CTEMP for use later. The Y register is then loaded with the value in COLOR. Next, a byte that contains a bit pattern corresponding to the color in COLOR is loaded, using Y as an index. This byte is AND'd with a mask from a lookup table of mask values, which is chosen using the X register as an index. The result will be a byte that has only those bits that correspond to the position and color of the point to be plotted set. This byte is exclusive OR'd with the byte previously stored into CTEMP. The Y register is once again cleared to 0 in preparation for an indirect addressing instruction. Using a store indirect instruction, the new byte, with the bits changed corresponding to the plotted point, is stored back into screen memory using the address in POINT. The last instruction in this subroutine is a return from subroutine instruction. The routine will destroy the original contents of the A, X, and Y registers.

The Equates Statements

The main lookup tables that were used by the point plotting routine have been assembled separately from the program. In order to make use of this data, the data file must be loaded into memory before the program is run. In order for the assembler to make use of data (or code) that is to be assembled at a different time, any names referring to code in the other file must be defined. The LOOKUP file includes the lookup tables described earlier, as well as a number set that can be used in scores. The addresses of where the data can be

found in memory have been defined with equate statements in the main program.

Defining the System

The next section of the main program code sets the parameters for the system. The Commodore 64's operating system is disabled, the keyboard is disabled, and the joysticks are configured as input devices. The border and background colors are set to black.

Bank 1 of RAM is selected as the graphics page. The base address of the high resolution bit map is set to an offset of $2000 in this bank of RAM, giving it an absolute address of $6000. This address is then stored in GBASE for use in the point plotting routine. The base address of the text page is set to an offset of $1C00, giving it an absolute address of $5C00. The screen mode is set to bit mapped graphics and the multicolor mode is selected. Color RAM is cleared to white, text RAM is cleared to $56 (green and purple), and the graphics page is cleared to $00.

Initializing RAM

Before the main program can be executed, all of the RAM variables must be set to their initial values. For example, the variable LIVES contains a number that represents how many player characters the player will start with Sound generators are enabled, the scores are cleared to 0, and starting positions are defined. Before starting the main loop, the collision registers are cleared, RASTER interrupts are enabled from the VIC chip on scan line $FB, and the interrupt disable bit is cleared.

THE MAIN PROGRAM LOOP

START marks the beginning of the main program loop. The first thing that needs to be done is to determine whether or not the game is still in play. This is done by ORing the LIVES register with all of the explosion registers. If the result is 0, the game is over. In this case, a jump to STWID, which is past the majority of the code, will be executed. The only other portion of the main program that will be executed is a check to see if the game should reset.

Scores

Since the screen has been set to be in a high resolution, multicolor bit map mode, the scores will have to be drawn from an appropriate character generator. The program will use a set of numbers that have been defined in the LOOKUP file. These characters will be drawn on the screen in the color as specified in the color RAM. Each number is comprised of 16 bytes.

The score routine is run only once every 16 screens. The timing is accomplished by loading SCREEN, ANDing it with $0F, and comparing the result to $0F. If the comparison is equal, the routine is executed. Otherwise, a jump is taken to NOSCR1, bypassing the routine. All of the timing elements in the program use this method to determine when to run a routine.

This routine prepares all four bytes of the score to be displayed. Each byte of SCORE contains two digits of the score. Each nibble of SCORE will be separated and stored in BUF through BUF+7. The high nibble will be shifted into the low nibble using the NIBLR macro. For the digits that are in the low nibble, the upper nibble will be masked to 0 using the AND #$0F instruction.

Once all of the digits have been place in BUF, the score will be run through a leading zero suppression routine. Although this is not essential for the proper operation of the game, it tends to make the game look more professional. A score without leading zeroes is also easier for the player to read. Leading zero suppression is accomplished by checking the most significant digit of the score first. If the byte is a 0, the byte is replaced with a $0A. When the score is displayed, a $0A will be displayed as a blank. This sequence will be repeated for the most significant 7 digits. If a digit other than a 0 is encountered, the program will branch to NZSUP, ending the suppression routine. This will prevent significant zeroes from being suppressed.

At this point, the score is ready to be displayed. To prepare for the display loop, the X register is cleared to 0, and the base address of the video screen ($6000) is stored in BUF1+2.

In the beginning of the loop, the most significant digit of the score is loaded into the accumulator

and shifted left by one bit. The byte is then transferred into the Y register, where it will be used as an index into a table of addresses that correspond to the numbers in the character table. The byte had to be shifted left because it was to be used as an index to a table of two-byte addresses, as opposed to a table of single-byte data. The address of the digit is stored in BUF1, and the Y register is cleared to 0 to prepare for the inner loop.

In the inner loop of the score routine, a byte is loaded indirectly from the address in BUF1 and stored indirectly at the address in BUF1+2. The Y register is then incremented and checked for $10. If Y is not yet $10, the routine loops to MKSCR2 and moves the next byte of the digit to the screen. When the digit is finished, a two byte increment of $10 is performed on BUF1+2. This increment changes the indirect address to point at the next character position on the screen. The X register is then incremented and compared to 8. If the X register is not yet 8, the routine loops to MKSCR1, which will move the next digit to the screen. When the X register reaches 8, the routine ends with all 8 digits displayed on the screen.

Moving the Bad Guys

This is the routine that moves the bad guys. At the top of the loop, WHOLIV is checked to see if the bad guy is still alive. The explosion register is then checked for 0 to verify that the bad buy is not in the explosion process. If the bad guy is alive, a byte is loaded using an address found in the appropriate MEANMV register. This byte is treated as a direction byte. If it is not $00, a branch is taken to the movement portion of the routine. The byte is stored in BUF1 for use later. The direction register is checked, and if the direction is 1, the direction bytes is inverted. The speed of movement is determined by using LEVEL as an index into SPEEDH and SPEEDY. These values are placed into the horizontal and vertical increment registers, HORNC and VERNC, which are used by the movement subroutines. Starting from the least significant bit, the lower four bits of the direction byte represent up, down, left, and right. The direction byte is rotated right, and if the carry bit is set as a result, the proper movement subroutine is called.

After checking all four bits of the direction register, the MEANMV register pair is incremented to point at the next instruction byte. The loop counters and index registers are then incremented, and the program branches back to the top of the loop. This sequence is repeated until all of the live bad guys have been moved.

Had the direction byte contained a $00, it would indicate the end of the movement table. In this case, a new movement table would be selected. The type of bad guy is determined by using LEVEL as an index into BADSEQ, which is a table of bad guy types. The type byte is then shifted left in preparation of use as an index into a base address table of movement patterns. This sequence allows specific types of movement to be used with specific types of bad guys. The routine then loads a random number that will determine which of the four possible movement patterns will be selected for the bad guy. The new address, which points to the movement pattern, is loaded and stored in the appropriate MEANMV register pair. The direction byte is then loaded and control is passed to the movement portion of the routine.

Incrementing the Level of Play

In order to advance to the next level of play, all of the bad guys have to be killed. WHOLIV contains the current status of the bad guys. Each of the first 7 bits of WHOLIV corresponds to one of the bad guys. When all of the bits are 0, all of the bad guys are dead. This is checked by loading WHOLIV into the accumulator, ANDing it with #$7F, and checking it for 0. If it isn't 0, the routine jumps to LEVDN, bypassing this routine.

When the level of play needs to be incremented, new vertical positions, as well as new horizontal positions, are chosen for all of the bad guys. The directions are cleared, and new sprite pointers are chosen. WHOLIV is reset to $FF. LEVEL is then incremented and checked for $12, which is the highest level defined. If you wish to define more levels of play, this number needs to be increased. If the level exceeds $12, LEVEL is decremented to $12.

Seeing If the Bad Guy Is Hit

The purpose of this routine is to determine whether or not the player's shot was hit by one of the bad guys, and if it was, to determine which bad guy hit the shot. The hardware collision registers can be used to determine whether or not a collision took place, but they cannot necessarily be used to determine which characters were hit. If two or more sprites collided, and the shot hit a sprite, you can not differentiate between the various collisions. Because the player and his shot are displayed on alternate screens, the collision status is only valid on even screens, and the routine will be bypassed on odd screens. If it is determined that a collision took place, collision checking will be done through the software. Software collision checking could be used instead of hardware collision checking, and in many cases is preferable, but a combination of the two will run faster. In the next section you will see a situation in which the hardware can be used exclusively for collision checking.

The temporary sprite collision register TMSCOL is loaded and checked for a collision in sprite 0. Sprite 0 is the player's shot. If it wasn't hit, there is no need to check any further, so a branch will be taken to BCOLND, passing over the routine.

Had the sprite been hit, WHOLIV would have been shifted left one bit to align its bits with the bits in TMSCOL. These two bytes are ANDed together and ANDed with $FE. This procedure will clear the shot collision bit. If the result is 0, any collisions were with dead bad guys and can be ignored. Otherwise, the result will be shifted to the right one bit in preparation for the software collision check loop. This value will be place in BUF for use in the following routine.

BUF is rotated right to check the least significant bit. Because this bit is rotated into the CARRY bit, it can be checked with a BCC instruction. If the carry bit is clear, there was no collision with this sprite, and a branch will be taken to the end of the loop. If the sprite was involved in a collision, its explosion register will be checked to see if the bad guy was already dead. At this point, after it has been determined that the bad guy is alive and has been involved in a collision, it must be whether or not the bad guy is near enough to the player's shot to have been hit. An offset is added to the bad guy's vertical position, and the shot vertical position is subtracted from it. The result is then compared to a value that corresponds to the height of the bad guy. If the carry bit is clear, the bad guy was within range of the shot vertically. The horizontal position is checked in the same way.

If both the horizontal and vertical positions of the bad guy are in range, the bad guy was hit. A value is then loaded into the appropriate MEANE explosion timer register. The shadow registers for voice channel 2 of the SID chip are initialized to the values that will cause an explosion. The number of points for the bad guy is then calculated and added into the score. Before the addition takes place, the decimal mode is set. After the new score is calculated, the decimal mode must be cleared. Failure to clear the decimal mode will lead to erratic operation of the program. Before exiting the routine, the player's shot is disabled so that it can't hit more than one bad guy.

Seeing If the Player Is Hit

Since the bad guys shots are drawn into the background and the player is a sprite, a collision can be detected between the two by checking the sprite to background collision register (SBCOL) in the VIC chip. During INTO, the data is transferred from the VIC chip into TMBCOL so that it will be available for use when needed. This routine checks bit 0 of TMBCOL and if it is set, initiates the explosion sequence for the player. Before the explosion sequence is initiated, the player explosion register is checked to make sure that the player is not already dead. The shadow registers for voice 1 are set to the proper values for an explosion, and LIVES is decremented. A value is stored into the explosion times to both force a delay before the next player appears and to allow time for the explosion to be seen.

Launching the Shots

Once very four screens, the program will try to launch a shot for the bad guys. The SHOTS register

is loaded, shifted right, and then ANDed with WHOLIV. The resulting byte had a bit set for every bad guy who is both alive and doesn't have a shot in flight. This byte is stored in BUF for use in the following loop.

In the shot launching loop, BUF is rotated right. If the CARRY bit is clear, the routine branches to the end of the loop. With the CARRY bit set, the routine has a chance to launch a shot. The Y register is loaded with LEVEL. The ACCUMULATOR is then loaded with a value from FIRPOW indexed by Y. This value is compared to RAND2. If RAND2 is less than the value from FIRPOW, the routine will try to launch a shot. As at last check before launching a shot, the appropriate explosion register is checked. If the bad guy is not exploding, the shot will be launched. The appropriate bit in SHOTS is set to indicate that the shot is in flight. Shot X and Y positions are taken form the bad guy. An offset is added into the vertical position before storing the value into the shot vertical register. A random number, which is masked and used as an index into a table of shot directions is loaded. This value is stored into the shot direction register. The shot X and Y coordinates are loaded into XPNT and YPNT in preparation for plotting the shot on the screen. A jump to the XPLOT subroutine, which will exclusively OR the shot to the screen, is then executed.

Voice channel 2 is set up to generate a tone to indicate that a shot has been launched. A frequency is chosen depending on the number of the bad guy who launched the shot. A higher tone will be generated for a higher bad guy number. After the sound generator is enabled, a jump is executed past the end of the loop.

Moving the Bad Guys' Shots

In order to move one of the bad guys' shots, it must first be unplotted from the old position and then replotted in the new position. SHOTS is used to determine whether or not a shot is in flight by storing it in BUF and rotating BUF to the right. If the carry bit is set, a shot is in flight.

The X and Y position of the shot is moved to XPNT and YPNT before the jump to the XPLOT subroutine. After the shot is unplotted, the shot direction is placed in BUF1. This byte is rotated right, and if the carry bit is set, the shot's position is incremented in the proper direction. The lower three bits of the shot direction register represent the directions up, down, and left starting from the least significant bit. A bad guy will never shoot to the right. The speed of the shot is determined by the SPEEDV and SPEEDH tables indexed by LEVEL.

Moving the Player's Shot

Moving the player's shot is much easier than moving the bad guys' shots. Since the player's shot is a sprite, it can be moved by changing its horizontal position. SHOTS is checked to see if a shot is in flight. If it is, SHOTS is incremented twice.

Checking the Shot Positions

The horizontal position of the player's shot is checked first. If the shot is in flight, and this check reveals that it has reached the right side of the screen, its bit in SHOTS is cleared.

The bad guys' shots must also be checked to see if they have reached the right side, the top, or the bottom. If the shot has reached one of the limits, the shot is unplotted, and its bit in SHOTS is cleared.

Animating the Sprites

All the sprites in this game use four sprite patterns to achieve animation. The sprite pattern is changed once every 16/60 of a second. The lower two bits of the sprite pointers determine which of the four patterns will be shown at any given time. This routine strips the lower two bits from the sprite pointer, increments them, and recombines the result with the pointer, pointing at the next sprite in the sequence.

Displaying the Number of Lives

This routine uses the same technique as the display score routine to display the number of lives remaining. The value in LIVES is shifted left and

used as an index into the table of number addresses. A short loop is then used to move the $10 bytes of data, which form a number, to the screen starting at $60C0.

Resetting the Game

In case you want to reset the game before it is over, this routine has been provided. It reads the value of joystick 2 and if the fire button is pressed, jumps to START.

Maintaining the Timing

In order to maintain consistent timing for the entire main loop, the program must wait at this point for an interrupt to occur. Since the interrupts occur every 1/60 of a second, they can be used to set the smallest timing unit. This routine works by loading ENABLE with $00. ENABLE is then loaded and checked until it is no longer $00. At this point, a jump is taken back to START, which is the top of the loop. ENABLE will be changed during the INTO interrupt routine.

THE INTERRUPT ROUTINES

INTO is the main interrupt routine for this program. When the program is initialized, a RASTER interrupt is set to occur on scan line $FB. This is the first line of border color at the bottom of the screen. At this point, any changes in the sprite and screen color registers will not be seen until the beam reaches the top of the visible portion of the screen. Since there is a large border at the top and the bottom of the screen, there is a lot of time to make changes to the various registers during this interval.

When the RASTER interrupt is generated, the C-64's operating system pushes the A, X, Y, and STATUS registers, as well as a return address, onto the stack. After the registers have been saved, an indirect jump is taken through CINV, which was initialized to point at INTO. On entry to the INTO interrupt routine, the sprite collision registers are transferred into their temporary registers, TMSCOL and TMBCOL. The system timers are updated next. SCREEN and RANSEC are incremented, and RANSEC is checked for $3C. If RANSEC equals $3C, it is cleared and SECOND is incremented.

After the timing is updated, the sprite control registers in the VIC chip are updated. The routine sets up a loop that will move the positions, colors, and sprite pointers for the mountain chain into the VIC chip. The MNTMSB register is moved into the XMSB register. All sprites are expanded in the X and Y direction, and the sprite multicolor mode is cleared. Since the beam has not yet reached the top of the visible portion of the screen, the mountain sprites will be all set up when the beam is in position to display them.

Some more timing is taken care of next. If the player explosion register is not 0, it is decremented. Similarly, SNDTM1 and SNDTM2 are decremented if they are not 0. When either one reaches 0, the release sequence is started in its corresponding voice channel. The shadow registers of the SID chip registers are next transferred into their corresponding registers.

A new random number is generated next. The lower two bits of SCREEN determine which of the four RAND registers will be updated with the new random number. Rather than simply transferring RANDOM into a RAND register, the random number is a combination of all of the RAND registers as well as RANDOM and RANSEC. This forces the random number to change no matter what frequency voice channel 3 is set to.

The Player Controls

Before the end of the INTO interrupt routine, the player's joystick is checked, and the player moved if necessary. LIVES is checked first to verify that the player is still alive. If he is, PLAYE is checked to make sure that the player is not exploding. The player is moved on alternate screens as a way to keep his speed reasonably slow. The joystick data is loaded and inverted, and the upper four bits are masked to 0. If the lower four bits equal $F, the joystick is centered and the movement routine is bypassed. If the joystick was moved, the data is put in IBUF, where it will be rotated to determine the direction of movement. When the joystick data is rotated right, the lower four bits represent up, down,

left, and right respectively. The player is moved by rotating the joystick data to the right, and for each bit set, moving the player by one in the appropriate direction. Rather than simply using an INC instruction, the player direction is changed using ADC and SBC instructions. This allows the new position to be checked to be sure that it is in bounds for the player before the new position is stored.

After the player position has been adjusted, the joystick data is checked to see if the fire button has been pressed. If it has, SHOTS is checked to see if the shot is already in flight. PLAYE is also checked to prevent the player from shooting if he is exploding. IF a shot can be fired, bit one of the SHOTS register is set to one, indicating a shot in flight. The player position is transferred into the shot position after adding an offset corresponding to the players gun. SHOTSD is set to $08, indicating that the shot should go to the right. Voice 1 of the SID chip is then initialized to a shooting sound.

Clear and Pull

At the end of the interrupt routine, the interrupt vector CINV is set to point at INT1. A RASTER interrupt is enabled to occur on scan line $54, which corresponds to the bottom of the mountains. A $01 is stored in ENABLE so that the main loop may exit its hold loop and execute the code once again. Use of the ENABLE register in this way prevents the main loop from running free, causing unpredictable timing in the program. The IPULL macro is used as the last instruction of the routine, clearing the video interrupts, restoring the A, X, and Y registers, and returning from the interrupt.

INT1

As this is another interrupt routine, all of the registers are pushed onto the stack before the routine is entered. Once in the routine, the sprite collision registers are cleared by loading them into the accumulator. After clearing the registers, 3 NOP instructions are executed. These are used as time delays to ensure that the beam is off the visible portion of the screen before the background color is changed to green. As in INTO, the sprite control registers in the VIC chip are updated with player and bad guy positions, colors, and sprite pointers. Since these horizontal positions are stored in an eight bit format, they must be expanded into nine bit values. This is accomplished by shifting the byte to the left and storing the byte in the sprite horizontal position register. If the CARRY bit is set by this operation, the corresponding bit is set in HMSB. In this case, HMSB is being used as a temporary register to generate a byte with all of the ninth bits of the sprite horizontal positions. After all of the new values have been loaded into the VIC chip, HMSB is transferred into the XMSB register.

The player and his shot are displayed using the same sprite. When the player's shot is enabled, the player will be displayed on even screens and the shot is displayed on odd screens. On odd screens, the shot position will be moved into the VIC registers, and the sprite pointer will be changed to point at a shot sprite. Bit 0 of XMSB is also changed accordingly.

The sprite X and Y multipliers are cleared, and the multicolor sprite mode is enabled for all sprites. WHOLIV is loaded and shifted left, and bit 0, indicating the player sprite, is set. The resulting byte has a bit set for each of the sprites to be displayed. This value is stored in the sprite enable register, SPREN.

Moving Mountains

On each screen, the mountains move one pixel to the left. When one of the mountain segment sprites leaves the visible portion of the screen on the left, it is moved past the visible portion of the screen on the right. In this way, the mountains appear to move smoothly and continuously. A loop is set up to check the mountain sprite positions. The sprite horizontal position is unpacked from a nine bit number with the ninth bit in MNTMSB to a two byte number. If the sprite position if $01E8, which is off the screen on the left, it is repositioned to $0170, which is off the screen to the right.

After the mountains have been moved, the explosion counters for the bad guys are checked. If one of the explosion timers is not 0, it is

decremented. When an explosion timer is decremented to 0, the appropriate bit in WHOLIV is cleared. This indicates that the bad guy should not be displayed any longer.

Before the end of the interrupt routine, CINV is set to point at INTO, which is the next interrupt routine to be executed. A raster interrupt is enabled to occur on scan line $FB, which is on the first scan line below the visible portion of the screen. Video interrupts are cleared, and the A, X, and Y registers are restored to their values before the interrupt. At this point, an RTI instruction is executed, returning program execution to the point where the interrupt was generated.

THE MOVEMENT SUBROUTINES

There are four subroutines that are used to adjust the bad guys' position registers. Each routine uses the appropriate ADC or SBC to change the value in the appropriate position register. Before it stores the new value into the position register, the new value is checked to be sure that it is in the proper range to be on the screen. These subroutines use the Y register as an index into the bad guy position registers. This allows the subroutines to be called from within movement loops without any preparation other than the use of the Y register as the loop counter.

AN END NOTE

The last instruction in the program before the .END assembler directive is a NOP instruction. Due to a bug in the Commodore assembler, the instruction counter does not always point to the last instruction in the file. This will usually occur when the last instruction in the file is a macro call. To prevent this from being a problem, the NOP instruction forces the instruction counter to point a standard OP CODE past the last instruction in program. Even if the assembler doesn't save the last byte of the program, it will only be an instruction that is not even a part of the program.

Appendix A

The entire instruction set for the 6510 microprocessor is described in this section. For each instruction, you are given the instruction name as well as a short description of what the instruction does. The STATUS bits affected by the instruction are shown, followed by a long description of the instructions' operation. Below these is a chart of the addressing mode options for the instruction, the code, format, the number of bytes the instruction requires, and the number of clock cycles the instruction requires to execute. For certain instructions, an asterisk appears in the column following the number of clock cycles. This indicates that you must add one to the number of cycles if the operation crosses a page boundary. The instructions are arranged alphabetically in this section.

ADC

Add memory to accumulator with carry.

N	Z	C	I	D	V
*	*	*			*

The data in the ACCUMULATOR is added to the immediate data or the data in the specified memory location. The CARRY bit is added into bit 0 of the result. If the result exceeds 8 bits, the CARRY bit will be set after the addition.

If the DECIMAL mode is set, the addition will be performed treating the data as binary coded decimal data. In this mode, the carry bit will be set for results over 99.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	69	ADC #OPER	2	2
ZERO PAGE	65	ADC OPER	2	3
ZERO PAGE,X	75	ADC OPER,X	2	4
ABSOLUTE	6D	ADC OPER	3	4
ABSOLUTE,X	7D	ADC OPER,X	3	4 *
ABSOLUTE,Y	79	ADC OPER,Y	3	4 *
(INDIRECT,X)	61	ADC (OPER,X)	2	6
(INDIRECT),Y	71	ADC (OPER),Y	2	5 *

AND

AND memory with accumulator.

N	Z	C	I	D	V
*	*				

A logical AND is performed between the data in the ACCUMULATOR and the immediate data or the data in the specified memory location.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	29	AND #OPER	2	2
ZERO PAGE	25	AND OPER	2	3
ZERO PAGE,X	35	AND OPER,X	2	4
ABSOLUTE	2D	AND OPER	3	4
ABSOLUTE,X	3D	AND OPER,X	3	4 *
ABSOLUTE,Y	39	AND OPER,Y	3	4 *
(INDIRECT,X)	21	AND (OPER,X)	2	6
(INDIRECT),Y	31	AND (OPER),Y	2	5

ASL Shift one bit left.

N Z C I D V
* * *

The contents of the ACCUMULATOR or the specified memory location are shifted left one bit. The most significant bit is shifted into the CARRY bit and the least significant bit is cleared.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ACCUMULATOR	0A	ASL A	1	2
ZERO PAGE	06	ASL OPER	2	5
ZERO PAGE,X	16	ASL OPER,X	2	6
ABSOLUTE	0E	ASL OPER	3	6
ABSOLUTE,X	1E	ASL OPER,X	3	7

BCC Branch on carry clear.

N Z C I D V

A branch is performed if the CARRY bit is clear.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	90	BCC OPER	2	2

BIT Test bits in memory with accumulator.

N Z C I D V
M_7 * M_6

The contents of the ACCUMULATOR are logically ANDed with the specified memory location, but the result of the operation is not stored. Bit 7 of the result is stored in the NEGATIVE bit of the STATUS register and bit 6 of the result is stored in the OVERFLOW bit of the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ZERO PAGE	24	BIT OPER	2	3
ABSOLUTE	2C	BIT OPER	3	4

BCS

Branch on carry set.

N Z C I D V

A branch is performed if the CARRY bit is set.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	B0	BCS OPER	2	2

BEQ

Branch on result zero.

N Z C I D V

A branch is performed if the ZERO bit is set.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	F0	BEQ OPER	2	2

BMI

Branch on result minus.

N Z C I D V

A branch is performed if the NEGATIVE bit is set.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	30	BMI OPER	2	2

BNE

Branch on result not zero.

N Z C I D V

A branch is performed if the ZERO bit is clear.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	D0	BNE OPER	2	2

BPL

Branch on result plus.

N Z C I D V

A branch is performed if the NEGATIVE bit is clear.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	10	BPL OPER	2	2

BRK

Force break.

N Z C I D V

The break command is used to initiate an interrupt under software control. This command can't be masked by the interrupt disable bit. By examining bit 4 of the status register, you can determine whether the interrupt routine was initiated through a break command or an external interrupt.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	00	BRK	1	7

BVC

Branch on overflow clear.

N Z C I D V

A branch is performed if the OVERFLOW bit is clear.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	50	BVC OPER	2	2

BVS

Branch on overflow set.

N Z C I D V

A branch is performed if the OVERFLOW bit is set.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
RELATIVE	70	BVS OPER	2	2

CLC

Clear carry bit.

N Z C I D V
0

CLC clears the CARRY bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	18	CLC	1	2

CLD

Clear decimal mode.

N	Z	C	I	D	V
				0	

CLD clears the DECIMAL mode bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	D8	CLD	1	2

CLI

Clear interrupt disable bit.

N	Z	C	I	D	V
			0		

CLI clears the INTERRUPT DISABLE bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	58	CLI	1	2

CLV

Clear overflow bit.

N	Z	C	I	D	V
					0

CLV clears the OVERFLOW bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	B8	CLV	1	2

CMP

Compare memory and accumulator.

N Z C I D V
***

The immediate data or the data in the specified memory location are subtracted from the ACCUMULATOR. The result is not stored, but the bits in the STATUS register are affected. The flags are affected as follows:

A	<	MEMORY	CARRY	=	CLEAR		
A	=	MEMORY	ZERO	=	SET CARRY	=	SET
A	>=	MEMORY	ZERO	=	CLEAR CARRY	=	SET

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	C9	CMP #OPER	2	2
ZERO PAGE	C5	CMP OPER	2	3
ZERO PAGE,X	D5	CMP OPER,X	2	4
ABSOLUTE	CD	CMP OPER	3	4
ABSOLUTE,X	DD	CMP OPER,X	3	4 *
ABSOLUTE,Y	D9	CMP OPER,Y	3	4 *
(INDIRECT,X)	C1	CMP (OPER,X)	2	6
(INDIRECT),Y	D1	CMP (OPER),Y	2	5 *

CPX

Compare memory and X register.

N Z C I D V
***

The immediate data or the data in the specified memory location are subtracted from the X REGISTER. The result is not stored, but the bits in the STATUS register are affected. The flags are affected as follows:

X	<	MEMORY	CARRY	=	CLEAR		
X	=	MEMORY	ZERO	=	SET CARRY	=	SET
X	>=	MEMORY	ZERO	=	CLEAR CARRY	=	SET

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	E0	CPX #OPER	2	2
ZERO PAGE	E4	CPX OPER	2	3
ABSOLUTE	EC	CPX OPER	3	4

CPY

Compare memory and Y register.

N Z C I D V
***

The immediate data or the data in the specified memory location are subtracted from the Y REGISTER. The result is not stored, but the bits in the STATUS register are affected. The flags are affected as follows:

Y <	MEMORY	CARRY	=	CLEAR		
Y =	MEMORY	ZERO	=	SET CARRY	=	SET
Y >=	MEMORY	ZERO	=	CLEAR CARRY	=	SET

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	C0	CPY #OPER	2	2
ZERO PAGE	C4	CPY OPER	2	3
ABSOLUTE	CC	CPY OPER	3	4

DEC

Decrement memory by one.

N Z C I D V
**

DEC decrements the specified memory location by one.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ZERO PAGE	C6	DEC OPER	2	5
ZERO PAGE,X	D6	DEC OPER,X	2	6
ABSOLUTE	CE	DEC OPER	3	6
ABSOLUTE,X	DE	DEC OPER,X	3	7

DEX

Decrement X register by one.

N Z C I D V
**

DEX decrements the contents of the X register by one.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	CA	DEX	1	2

DEY

Decrement Y register by one.

N Z C I D V
**

DEY decrements the contents of the Y register by one.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	88	DEY	1	2

EOR

Exclusive-OR memory with accumulator.

N Z C I D V
**

EOR performs an exclusive OR between the ACCUMULATOR and the immediate data or the specified memory address.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	49	EOR #OPER	2	2
ZERO PAGE	45	EOR OPER	2	3
ZERO PAGE,X	55	EOR OPER,X	2	4
ABSOLUTE	4D	EOR OPER	3	4
ABSOLUTE,X	5D	EOR OPER,X	3	4 *
ABSOLUTE,Y	59	EOR OPER,Y	3	4 *
(INDIRECT,X)	41	EOR (OPER,X)	2	6
(INDIRECT),Y	51	EOR (OPER),Y	2	5 *

INC

Increment memory by one.

N Z C I D V
*** ***

INC increments the specified memory address by one.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ZERO PAGE	E6	INC OPER	2	5
ZERO PAGE,X	F6	INC OPER,X	2	6
ABSOLUTE	EE	INC OPER	3	6
ABSOLUTE,X	FE	INC OPER,X	3	7

INX

Increment X register by one.

N Z C I D V
*** ***

INX increments the contents of the X register by one.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	E8	INX	1	2

INY

Increment Y register by one.

N Z C I D V
*** ***

INY increments the contents of the Y register by one.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	C8	INY	1	2

JMP

Jump to new location.

N Z C I D V

JMP changes the address in the program counter. Program execution continues at the new address.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ABSOLUTE	4C	JMP OPER	3	3
INDIRECT	6C	JMP (OPER)	3	5

JSR

Jump to subroutine.

N Z C I D V

JSR changes the address in the program counter after pushing a return address onto the stack. Program execution continues at the new address.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ABSOLUTE	20	JSR OPER	3	6

LDA

Load accumulator.

N Z C I D V
* *

LDA loads the ACCUMULATOR with data from the specified memory location. If the IMMEDIATE mode is specified, the data to be loaded will be the second byte of the instruction.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	A9	LDA #OPER	2	2
ZERO PAGE	A5	LDA OPER	2	3
ZERO PAGE,X	B5	LDA OPER,X	2	4
ABSOLUTE	AD	LDA OPER	3	4
ABSOLUTE,X	BD	LDA OPER,X	3	4 *
ABSOLUTE,Y	B9	LDA OPER,Y	3	4 *
(INDIRECT,X)	A1	LDA (OPER,X)	2	6
(INDIRECT),Y	B1	LDA (OPER),Y	2	5 *

LDX

Load X register.

N Z C I D V
*** ***

LDX loads the X REGISTER with data from the specified memory location. If the IMMEDIATE mode is specified, the data to be loaded will be the second byte of the instruction.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	A2	LDX #OPER	2	2
ZERO PAGE	A6	LDX OPER	2	3
ZERO PAGE,Y	B6	LDX OPER,Y	2	4
ABSOLUTE	AE	LDX OPER	3	4
ABSOLUTE,Y	BE	LDX OPER,Y	3	4 *

LDY

Load Y register.

N Z C I D V
*** ***

LDY loads the Y REGISTER with data from the specified memory location. If the IMMEDIATE mode is specified, the data to be loaded will be the second byte of the instruction.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	A0	LDY #OPER	2	2
ZERO PAGE	A4	LDY OPER	2	3
ZERO PAGE,X	B4	LDY OPER,Y	2	4
ABSOLUTE	AC	LDY OPER	3	4
ABSOLUTE,X	BC	LDY OPER,Y	3	4 *

LSR

Shift right one bit (memory or accumulator).

N Z C I D V
0 * *

LSR shifts the ACCUMULATOR or the specified memory location to the right one bit. A 0 is shifted into the most significant bit, clearing the negative bit in the STATUS register. The least significant bit is shifted into the CARRY bit.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ACCUMULATOR	4A	LSR A	1	2
ZERO PAGE	46	LSR OPER	2	5
ZERO PAGE,X	56	LSR OPER,X	2	6
ABSOLUTE	4E	LSR OPER	3	6
ABSOLUTE,X	5E	LSR OPER,X	3	7

NOP

No operation.

N Z C I D V

This instruction doesn't do anything for 2 cycles.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	EA	NOP	1	2

ORA

OR memory with accumulator.

N Z C I D V
*** ***

ORA performs a logical OR between the ACCUMULATOR and the immediate data or the data in the specified memory location.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	09	ORA #OPER	2	2
ZERO PAGE	05	ORA OPER	2	3
ZERO PAGE,X	15	ORA OPER,X	2	4
ABSOLUTE	0D	ORA OPER	3	4
ABSOLUTE,X	1D	ORA OPER,X	3	4 *
ABSOLUTE,Y	19	ORA OPER,Y	3	4 *
(INDIRECT,X)	01	ORA (OPER,X)	2	6
(INDIRECT),Y	11	ORA (OPER),Y	2	5

PHA

Push accumulator on to stack.

N Z C I D V

PHA pushes the ACCUMULATOR onto the stack.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	48	PHA	1	3

PHP

Push status register on to stack.

N Z C I D V

PHP pushes the STATUS byte onto the stack.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	08	PHP	1	3

PLA

Pull accumulator from stack.

N Z C I D V
* *

PLA pulls the ACCUMULATOR off the stack.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	68	PLA	1	4

PLP

Pull status register from stack.

N Z C I D V
from stack

PLP pulls the STATUS byte off the stack.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	28	PLP	1	4

ROL

Rotate memory or accumulator one bit left.

N Z C I D V
* * *

ROL rotates the ACCUMULATOR or the specified memory location one bit to the left through the CARRY bit. The CARRY bit is rotated into the least significant bit and the most significant bit is rotated into the CARRY bit.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ACCUMULATOR	2A	ROL #OPER	1	2
ZERO PAGE	26	ROL OPER	2	5
ZERO PAGE,X	36	ROL OPER,X	2	5
ABSOLUTE	2E	ROL OPER	3	6
ABSOLUTE,X	3E	ROL OPER,X	3	7

ROR

Rotate memory or accumulator one bit right.

N Z C I D V
* * *

ROR rotates the ACCUMULATOR or the specified memory location one bit to the right through the CARRY bit. The CARRY bit is rotated into the most significant bit and the least significant bit is rotated into the CARRY bit.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ACCUMULATOR	6A	ROR #OPER	1	2
ZERO PAGE	66	ROR OPER	2	5
ZERO PAGE,X	76	ROR OPER,X	2	6
ABSOLUTE	6E	ROR OPER	3	6
ABSOLUTE,X	7E	ROR OPER,X	3	7

RTI

Return from interrupt.

N Z C I D V
from stack

RTI returns from an interrupt by pulling the STATUS byte off the stack and pulling the PROGRAM COUNTER off the stack. The values pulled from the stack were pushed onto the stack by the interrupt or BRK instruction.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	40	RTI	1	6

RTS

Return from subroutine.

N Z C I D V

RTS returns from a subroutine to the instruction following the jump to subroutine instruction.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	60	RTS	1	6

SBC

Subtract memory from accumulator with carry.

N	Z	C	I	D	V
*	*	*			*

SBC subtracts the immediate data or the specified memory location from the ACCUMULATOR using the CARRY bit as a borrow bit. This instruction will subtract the complement of the CARRY bit from the least significant bit of the ACCUMULATOR.

If the DECIMAL mode is set, the data will be treated as binary coded decimal data. In this case, 99 is the highest value that can be represented in the ACCUMULATOR. As with the binary mode of operation, the CARRY bit will be cleared if the value goes below zero.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMMEDIATE	E9	SBC #OPER	2	2
ZERO PAGE	E5	SBC OPER	2	3
ZERO PAGE,X	F5	SBC OPER,X	2	4
ABSOLUTE	ED	SBC OPER	3	4
ABSOLUTE,X	FD	SBC OPER,X	3	4 *
ABSOLUTE,Y	F9	SBC OPER,Y	3	4 *
(INDIRECT,X)	E1	SBC (OPER,X)	2	6
(INDIRECT),Y	F1	SBC (OPER),Y	2	5 *

SEC

Set carry bit.

N	Z	C	I	D	V
		1			

SEC sets the CARRY bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	38	SEC	1	2

SED

Set decimal mode.

N	Z	C	I	D	V
				1	

SED sets the DECIMAL mode bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	F8	SED	1	2

SEI

Set interrupt disable bit.

N	Z	C	I	D	V
			1		

SEI sets the INTERRUPT DISABLE bit in the STATUS register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	78	SEI	1	2

STA

Store accumulator in memory.

N Z C I D V

STA stores the contents of the ACCUMULATOR into the specified memory location.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ZERO PAGE	85	STA OPER	2	3
ZERO PAGE,X	95	STA OPER,X	2	4
ABSOLUTE	8D	STA OPER	3	4
ABSOLUTE,X	9D	STA OPER,X	3	5
ABSOLUTE,Y	99	STA OPER,Y	3	5
(INDIRECT,X)	81	STA (OPER,X)	2	6
(INDIRECT),Y	91	STA (OPER),Y	2	6

STX

Store X register in memory.

N Z C I D V

STX stores the contents of the X REGISTER into the specified memory location.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ZERO PAGE	86	STX OPER	2	3
ZERO PAGE,Y	96	STX OPER,Y	2	4
ABSOLUTE	8E	STX OPER	3	4

STY

Store Y register in memory.

N Z C I D V

STY stores the contents of the Y REGISTER into the specified memory location.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
ZERO PAGE	84	STY OPER	2	3
ZERO PAGE,X	94	STY OPER,X	2	4
ABSOLUTE	8C	STY OPER	3	4

TAX

Transfer accumulator to X register.

N Z C I D V
*** ***

TAX transfers the data in the ACCUMULATOR to the X register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	AA	TAX	1	2

TAY

Transfer accumulator to Y register.

N Z C I D V

TAY transfers the data in the ACCUMULATOR to the Y register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	A8	TAY	1	2

TSX

Transfer stack pointer to X register.

N Z C I D V
* *

TSX transfers the current STACK POINTER to the X register.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	BA	TSX	1	2

TXS

Transfer X register to stack pointer.

N Z C I D V

TXS transfers the data in the X register to the STACK POINTER.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	9A	TXS	1	2

TXA

Transfer X register to accumulator.

N Z C I D V
* *

TXA transfers the data in the X register to the ACCUMULATOR.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	8A	TXA	1	2

TYA

Transfer Y register to accumulator.

N Z C I D V
* *

TYA transfers the data in the Y register to the ACCUMULATOR.

Addressing Mode	OP Code	Instruction Format	No. Bytes	No. Cycles
IMPLIED	98	TYA	1	2

Appendix B

Descriptions of all the macros in the MACLIB file are in this section. Each description includes the macro name, the 6510 registers that are affected by calling the macro, a description of what the macro does, and the syntax of the instruction line. If a macro requires any parameters, they are described also. Most of the macros have been discussed elsewhere in the text of the book, and this section provides a quick reference guide to them. The macros are arranged alphabetically.

ADRES

REGISTERS DESTROYED

A	X	Y	SP
*			

ADRES loads a two byte address into memory locations.

```
ADRES  ?1,?2

?1=Address to be loaded into memory.
?2=Destination address of the address.
```

ANOP

REGISTERS DESTROYED

A	X	Y	SP

This macro produces no code, but provides a convenient place to hang a label.

```
ANOP
```

ASL2

REGISTERS DESTROYED

A	X	Y	SP
*			

ASL2 shifts a two byte value 1 bit to the left.

```
ASL2  ?1

?1=Address of the first of two bytes to shift left.
```

BANK

REGISTERS DESTROYED

A	X	Y	SP
*			

BANK selects the 16K bank of memory that the VIC chip can access.

```
BANK   ?1

?1=Number of the bank to select.

           0=$0000-$3FFF
           1=$4000-$7FFF
           2=$8000-$BFFF
           3=$C000-$FFFF
```

BGT

REGISTERS DESTROYED

A	X	Y	SP

A branch is performed if the CARRY bit is set and the ZERO bit is set.

```
BGT   ?1

?1=Destination address for the jump.
```

BLE

REGISTERS DESTROYED

A	X	Y	SP

A branch is performed if the CARRY bit is clear or the ZERO bit is set.

```
BLE   ?1

?1=Destination address for the jump.
```

BLNK

REGISTERS DESTROYED

A	X	Y	SP
*			

BLNK blanks the screen to the border color.

```
BLNK
```

DBADC

REGISTERS DESTROYED

A	X	Y	SP
*			

DBADC adds one two byte value to another.

```
DBADC  ?1,?2,?3

?1=Address of the first number.
?2=Address of the second number.
?3=Destination address of result.
```

DBDEC

REGISTERS DESTROYED

A	X	Y	SP
*			

DBDEC decrements a two byte number by a specified value.

```
DBDEC  ?1,?2

?1=Address of the number to be decremented.
?2=Number to be subtracted.
```

DBADCI

REGISTERS DESTROYED

A	X	Y	SP
*			

DBADCI adds a one byte number to a two byte number.

```
DBADCI  ?1,?2,?3

?1=Address of the first number.
?2=Immediate data or the address of the number to add.
?3=Destination address of result.
```

DBINC

REGISTERS DESTROYED

A	X	Y	SP
*			

DBINC increments a two byte number by a specified value.

```
DBINC  ?1,?2

?1=Address of the number to increment.
?2=Number to be added.
```

DBPL

REGISTERS DESTROYED

A	X	Y	SP

DBPL decrements the X or Y register and branches if the NEGATIVE bit is clear.

```
DBPL  ?1,?2 .

?1=X or Y to specify the register to be used.
?2=Destination of the branch

NOTE--Either the contents of the X register or the Y
register will be altered depending on ?1.
```

DBSBC

REGISTERS DESTROYED

A	X	Y	SP
*			

DBSBC subtracts one two byte value from another.

```
DBSBC  ?1,?2,?3

?1=Address of the first number.
?2=Address of the number to subtract.
?3=Destination address of result.
```

DBSBCI

REGISTERS DESTROYED

A	X	Y	SP
*			

DBSBCI subtracts a one byte number from a two byte number.

```
DBSBCI  ?1,?2,?3

?1=Address of the first number.
?2=Immediate data or the address of the number
   to subtract.
?3=Destination address of result.
```

DS

REGISTERS DESTROYED

A	X	Y	SP

DS defines a number of bytes to be allocated for data storage starting at the current program counter location.

```
DS  ?1

?1=Number of bytes to allocate. This macro should be
   preceded by a label.
```

FILBYT

REGISTERS DESTROYED

A	X	Y	SP
*		*	

FILBYT fills up to 255 bytes of memory with the same value.

```
FILBYT   ?1,?2,?3

?1=Address of the first byte to fill.
?2=Data to load into memory.
?3=Number of bytes to fill.
```

FILL

REGISTERS DESTROYED

A	X	Y	SP
*	*	*	

FILL fills a number of 256 byte pages of memory with a value.

```
FILL ?1,?2,?3

?1=Address of the first byte to fill.
?2=Data to fill the area with.
?3=Number of 256 byte pages to fill.
```

GRABAS

REGISTERS DESTROYED

A	X	Y	SP
*			

GRABAS sets the base address of graphics in the current bank.

```
GRABAS   ?1

?1=The base address of graphics memory. This address
   must be a multiple of $0800.
```

GRAPH

REGISTERS DESTROYED

A	X	Y	SP
*			

GRAPH enables a bit mapped graphics mode.

```
GRAPH
```

HDEC

REGISTERS DESTROYED

A	X	Y	SP
*	*		

HDEC decrements a nine bit sprite position with the ninth bit in HMSB. The X register must contain the index number of the byte to decrement in an 8 byte array of horizontal positions.

```
HDEC  ?1

?1=Base address of the horizontal position array.
```

HINC

REGISTERS DESTROYED

A	X	Y	SP
*	*		

HINC increments a nine bit sprite position with the ninth bit in HMSB. The X register must contain the index number of the byte to increment in an 8 byte array of horizontal positions.

```
HINC  ?1

?1=Base address of the horizontal position array.
```

IPULL

REGISTERS DESTROYED

A	X	Y	SP
*	*	*	

IPULL restores the A, X, and Y registers to their original values before an interrupt routine and clears the video interrupt.

```
IPULL
```

JCC

REGISTERS DESTROYED

A	X	Y	SP

JCC causes a jump if the CARRY bit is clear.

```
JCC   ?1

?1=Destination address for the jump.
```

JCS

REGISTERS DESTROYED

A	X	Y	SP

JCS causes a jump if the CARRY bit is set.

```
JCS   ?1

?1=Destination address for the jump.
```

JEQ

REGISTERS DESTROYED

A	X	Y	SP

JEQ causes a jump if the ZERO bit is set.

```
JEQ   ?1

?1=Destination address for the jump.
```

JGE

REGISTERS DESTROYED

A	X	Y	SP

JGE causes a jump if the CARRY bit is set.

```
JGE   ?1

?1=Destination address for the jump.
```

JGT

REGISTERS DESTROYED

A	X	Y	SP

JGT causes a jump if the ZERO bit is clear and the CARRY bit is set.

```
JGT   ?1

?1=Destination address for the jump.
```

JLE

REGISTERS DESTROYED A X Y SP

JLE causes a jump if the CARRY bit is clear and the ZERO bit is set.

```
JLE   ?1

?1=Destination address for the jump.
```

JLT

REGISTERS DESTROYED A X Y SP

JLT causes a jump if the CARRY bit is clear.

```
JLT   ?1

?1=Destination address for the jump.
```

JMI

REGISTERS DESTROYED A X Y SP

JMI causes a jump if the NEGATIVE bit is set.

```
JMI   ?1

?1=Destination address for the jump.
```

JNE

REGISTERS DESTROYED

A	X	Y	SP

JNE causes a jump if the ZERO bit is clear.

```
JNE   ?1

?1=Destination address for the jump.
```

JPL

REGISTERS DESTROYED

A	X	Y	SP

JPL causes a jump if the NEGATIVE bit is clear.

```
JPL   ?1

?1=Destination address for the jump.
```

LDMEM

REGISTERS DESTROYED

A	X	Y	SP
*			

LDMEM moves data from one location to another or loads an immediate value into memory using the accumulator.

```
LDMEM   ?1,?2

?1=Immediate data or memory location to move.
?2=Destination address.
```

LDMEMX

REGISTERS DESTROYED

A	X	Y	SP
	*		

LDMEMX moves data from one location to another or loads an immediate value into memory using the X register.

```
LDMEMX   ?1,?2

?1=Immediate data or memory location to move.
?2=Destination address.
```

LDMEMY

REGISTERS DESTROYED

A	X	Y	SP
		*	

LDMEMY moves data from one location to another or loads an immediate value into memory using the Y register.

```
LDMEMY   ?1,?2

?1=Immediate data or memory location to move.
?2=Destination address.
```

LSR2

REGISTERS DESTROYED

A	X	Y	SP
*			

LSR2 shift a two-byte value 1 bit to the right.

```
LSR2   ?1

?1=Address of the first of two bytes to shift right.
```

KILL

REGISTERS DESTROYED

A	X	Y	SP
*	*		*

KILL turns off the Commodore's operating system:
1. BASIC is disconnected.
2. Joysticks are enabled for input.
3. System timers are turned off.
4. Raster interrupts are enabled.
5. All pending interrupts are cleared.
6. Page 0 of RAM is cleared.
7. The NMI vector is set to point at an RTI instruction.

```
KILL
```

MULTOF

REGISTERS DESTROYED

A	X	Y	SP
*			

MULTOF disables a multicolor graphics mode.

```
MULTOF
```

MULTON

REGISTERS DESTROYED

A	X	Y	SP
*			

MULTON enables a multicolor graphics mode.

```
MULTON
```

MVCOL

REGISTERS DESTROYED

A	X	Y	SP
*	*	*	

MVCOL fills color RAM with a value.

```
MVCOL  ?1

?1=Value to fill color RAM with.
```

MVMEM

REGISTERS DESTROYED

A	X	Y	SP
*	*	*	

Moves a number of 256 byte pages of memory to a different area in memory.

```
MVMEM  ?1,?2,?3

?1=Starting address of memory to move.
?2=Destination address of data.
?3=Number of 256 byte pages to move.
```

NIBLL

REGISTERS DESTROYED

A	X	Y	SP
*			

NIBLL shifts the lower byte of the accumulator into the upper byte.

```
NIBLL
```

NIBLR

REGISTERS DESTROYED

A	X	Y	SP
*			

NIBLR shifts the upper byte of the accumulator into the lower byte.

```
NIBLR
```

NOT

REGISTERS DESTROYED

A	X	Y	SP
*			

NOT complements the contents of the accumulator.

```
NOT
```

PUNPCK

REGISTERS DESTROYED

A	X	Y	SP
*			

PUNPCK unpacks a nine bit horizontal position into a two byte number. The X register must contain the index number of the byte to unpack in an 8 byte array of horizontal positions.

```
PUNPCK  ?1,?2,?3

?1=Base address of the horizontal position array.
?2=Address of the most significant bit register.
?3=Destination of the unpacked bytes.
```

QDDEC

REGISTERS DESTROYED

A	X	Y	SP
*			

QDDEC decrements a four byte number by a specified value.

```
QDDEC  ?1,?2

?1=Address of the number to be decremented.
?2=Number to be subtracted.
```

QDINC

REGISTERS DESTROYED

A	X	Y	SP
*			

QDINC increments a four byte number by a specified value.

```
QDINC  ?1,?2

?1=Address of the number to increment.
?2=Number to be added.
```

RAST

REGISTERS DESTROYED

A	X	Y	SP
*			

RAST enables the VIC chip to generate an interrupt on a specified scan line.

RAST ?1

?1=Immediate data or memory location that specifies the scan line on which to generate an interrupt.

SMNPC

REGISTERS DESTROYED

A	X	Y	SP
*			

SMNPC unpacks a nine bit horizontal position into a one byte number. The X register must contain the index number of the byte to unpack in an 8 byte array of horizontal positions. The resulting single byte is the nine bit horizontal position divided by 2.

```
SMNPC  ?1,?2,?3

?1=Address of the horizontal position register.
?2=Address of the most significant bit register.
?3=Destination of the unpacked byte.
```

SMNPCX

REGISTERS DESTROYED

A	X	Y	SP
*			

SMNPCX unpacks a nine bit horizontal position into a one byte number. The X register must contain the index number of the byte to unpack in an 8 byte array of horizontal positions. The resulting single byte is the nine bit horizontal divided by 2.

```
SMNPCX  ?1,?2,?3

?1=Base address of the horizontal position array.
?2=Address of the most significant bit register.
?3=Destination of the unpacked byte.
```

TEXT

REGISTERS DESTROYED

A	X	Y	SP
*			

TEXT enables a character graphics mode.

```
TEXT
```

TPDEC

REGISTERS DESTROYED

A	X	Y	SP
*			

TPDEC decrements a three byte number by a specified value.

```
TPDEC  ?1,?2

?1=Address of the number to be decremented.
?2=Number to be subtracted.
```

TPINC

REGISTERS DESTROYED

A	X	Y	SP
*			

TPINC increments a three byte number by a specified value.

```
TPINC  ?1,?2

?1=Address of the number to increment.
?2=Number to be added.
```

TXBAS

REGISTERS DESTROYED

A	X	Y	SP
*			

TXBAS sets the base address of TEXT memory in the current bank.

```
TXBAS      ?1

?1=Base address of TEXT. Must be a multiple of $0400.
```

UNBLNK

REGISTERS DESTROYED

A	X	Y	SP
*			

UNBLNK enables the screen display.

```
UNBLNK
```

UNPACK

REGISTERS DESTROYED

A	X	Y	SP
*			

UNPACK unpacks a nine bit horizontal position into a two byte number. The X register must contain the index number of the byte to unpack in an 8 byte array of horizontal positions.

```
UNPACK  ?1,?2,?3

?1=Address of the horizontal position register.
?2=Address of the most significant bit register.
?3=Destination of the unpacked bytes.
```

Appendix C

In this section, you will find listings of all the source code files mentioned in the book. Memory dumps are provided for the binary files, such as sprites and lookup charts. Memory dumps are also provided for the assembled versions of the programs described in this book.

If you are going to enter one of the binary files by hand, you must be sure to enter the data exactly as shown. Otherwise, the program may not run, or in the case of graphics, will not look right.

Table C-1 is a list of the programs in this Appendix. The type of each program and its loading address are also provided. A program can be one of the following types:

Listing Number	Name	Type	Address	BASADR
C - 1	MACLIB	SRC		
C - 2	SYSDEF	SRC		
C - 3	INTER-DEMO	SRC		
C - 4	INT DEMO.O	BIN	$1000	4096
C - 5	SOUND	SRC		
C - 6	SOUND DEMO	BIN	$1000	4096
C - 7	COMMON	SRC		
C - 8	SNDDEF	SRC		
C - 9	DATA	SRC		
C - 10	EDIT SND	SRC		

Listing Number	Name	Type	Address	BASADR
C - 11	SOUND EDIT	BIN	$1000	4096
C - 12	KO-COM	BAS		
C - 13	COM-KO	BAS		
C - 14	DISPLAY PIC	BAS		
C - 15	MVIT	BIN	$2000	8192
C - 16	PIC A CASTLE	BIN	$6000	
C - 17	SPRITE MAKER	BAS		
C - 18	SLIB.O	BIN	$8000	
C - 19	CLSP2	BIN	$4000	
C - 20	SCREEN-MAKE	BAS		
C - 21	CLBACK1	BIN	$6000	
C - 22	CLSP1	BIN	$4000	
C - 23	PHOENIX V1.4N	BIN	$8000	32768
C - 24	BOGHOP	SRC		
C - 25	BOGHOP.O	BIN	$1400	5120
C - 26	BOGDEF	SRC		
C - 27	BOGDAT	SRC		
C - 28	XXPLOT	SRC		
C - 29	LOOKUP	BIN	$1000	
C - 30	BOGSPR	BIN	$4000	
C - 31	OOPLOT	SRC		

SRC An assembly language source code file. This type of file should be loaded by an assembler.

BAS A BASIC program. This type of file can be loaded by entering LOAD"NAME",8. The program can then be RUN.

BIN A binary data file. This type of file can be loaded by entering LOAD"NAME",8,1 or by a machine language monitor. If the file is a program, it can be run by entering SYS BASADR, where BASADR is the decimal address of where the program will be loaded. If the file is not a program, no BASADR will be listed.

A short description of each of the files follows.

Listing C-1, MACLIB, is the macro library source code.
Listing C-2, SYSDEF, is a file of system definitions.
Listing C-3, INTER-DEMO, is the source code for INT DEMO.O, a demonstration of the use of interrupts to change background colors.
Listing C-4, INT DEMO.O is the assembled version of INTER-DEMO.
Listing C-5, SOUND, is the source code file for the SOUND DEMO program.
Listing C-6, SOUND DEMO, is the assembled SOUND file, which can be run.
Listing C-7, COMMON, is a file of musical definitions.

Listing C-8, SNDDEF, is a file of RAM definitions used by both SOUND and EDIT SOUND.
Listing C-9, DATA, is a data file used by SOUND.
Listing C-10, EDIT SND, is the source code for the SOUND EDIT program.
Listing C-11, SOUND EDIT, is a screen editor for the SID chip. With this program, you can test sound effects in your program later.
Listing C-12, KO-COM translates a Koala Pad picture filename into a more usable format.
Listing C-13, COM-KO, translates a Koala Pad picture filename in the Commodore format to the Koala Pad format.
Listing C-14, DISPLAY PIC, will display a Koala Pad picture on the screen if its name has been translated to the Commodore format. This program can display PIC A CASTLE.
Listing C-15, MVIT, is a machine language subroutine used by DISPLAY PIC. This routine will display a picture once it is loaded into the computer.
Listing C-16, PIC A CASTLE, is a Koala Pad picture file whose name has been translated to the Commodore format.
Listing C-17, SPRITE MAKER, is a utility program that aids in the design of sprites.
Listing C-18, SLIB.O contains some machine language routines that are called by the SPRITE MAKER program.
Listing C-19, CLSP2, contains sprites used by the SPRITE MAKER program.
Listing C-20, SCREEN-MAKE, is a utility program that is designed to aid in the design of character graphics screens.
Listing C-21, CLBACK1, is a file that has the character set to be used by the SCREEN-MAKE program. You can add to this file or replace it with a character set of your own. If you replace this file, your new file must load into memory starting at $6000.
Listing C-22, CLSP1, is a sprite used by the SCREEN-MAKE program.
Listing C-23, PHOENIX V1.4N, is a ready-to-run arcade style game.
Listing C-24, BOGHOP, is the main code file for the game BOGHOP.
Listing C-25, BOGHOP.O, is a ready-to-run arcade game that is fully described in Chapter 12.
Listing C-26, BOGDEF, is a file of RAM variables used by the BOGHOP game.
Listing C-27, BOGDAT, is a data file used by the BOGHOP game.
Listing C-28, XXPLOT, is a point plotting subroutine. The LOOKUP file must be loaded and the variables defined, as in the beginning of the BOGHOP program, for this routine to work. Points are exclusively OR'd to the screen by this program.
Listing C-29, LOOKUP, is a data file that is used by XXPLOT and OOPLOT.
Listing C-30, BOGSPR, is a file of sprites used by the BOGHOP program.
Listing C-31, OOPLOT, is a point plotting routine similar to XXPLOT. The only difference is that OOPLOT ORs as point to the screen instead of exclusively ORing it.

Listing C-1: The MACLIB Source Code

```
1000 ;PUT"@0:MACLIB
1010 ;
1020 ;THIS IS A MACRO LIBRARY
1030 ;IT SHOULD BE THE FIRST LIBRARY
1040 ;LOADED
1050 ;
1060 ;LATEST ADDITIONS 03/16/84
1070 ;
```

```
1080 ;
1090 ;
1100         .MAC IPULL
1110         LDA #$FF
1120         STA VIRQ
1130         PLA
1140         TAY
1150         PLA
1160         TAX
1170         PLA
1180         RTI
1190         .MND
1200 ;
1210         .MAC KILL        ;KILLS O/S
1220         SEI
1230         LDMEM #$06,PORT ;DISCONNECT BASIC
1240         LDA #$00
1250         STA CIANT1
1260         STA CIANT2
1270         LDA CIANT1
1280         LDA CIANT2
1290         LDA #$00
1300         STA $DC02        ;JOYSTICK ENABLE
1310         STA $DC03
1320         STA $DC0E        ;KILL TIMERS
1330         STA $DC0F
1340         STA $DD0E
1350         STA $DD0F
1360         LDMEM #$01,VIRQM ;RAST INT ONLY
1370         LDMEM #$FF,VIRQ ;CLEAR VID INTS
1380         LDX #$02
1390         LDA #$00
1400 ?4      STA $00,X        ;CLEAR PAGE 0
1410         INX
1420         BNE ?4
1430 ;
1440         LDX #$FF
1450         TXS
1460         ADRES RET,NMINV
1470 ;
1480         .MND
1490 ;
1500 ;
1510         .MAC ADRES       ;SOURCE,DEST
1520         LDA #<?1
1530         STA ?2
1540         LDA #>?1
1550         STA ?2+1
1560         .MND
1570 ;
1580         .MAC DBINC       ;TWO BYTE INC
1590         LDA ?1
1600         CLC
1610         ADC #?2
1620         STA ?1
1630         LDA #$00
1640         ADC ?1+1
1650         STA ?1+1
1660         .MND
```

```
1670 ;
1680        .MAC DBDEC      ;2 BYTE DEC
1690        LDA ?1
1700        SEC
1710        SBC #?2
1720        STA ?1
1730        LDA ?1+1
1740        SBC #$00
1750        STA ?1+1
1760        .MND
1770 ;
1780        .MAC RAST
1790        LDA ?1
1800        STA RASTER
1810        LDA YSCRL
1820        AND #$7F
1830        STA YSCRL
1840        .MND
1850 ;
1860 ;
1870        .MAC DS
1880        *=*+?1
1890        .MND
1900 ;
1910 ;
1920        .MAC HINC       ;MSB OFFSET IN X
1930        INC ?1,X
1940        BNE ?6
1950        LDA HMSB
1960        ORA BITPOS,X
1970        STA HMSB
1980 ?6     .MND
1990 ;
2000 ;
2010        .MAC HDEC       ;MSB OFFSET IN X
2020        DEC ?1,X
2030        LDA ?1,X
2040        CMP #$FF
2050        BNE ?6
2060        LDA HMSB
2070        AND BITPOS,X
2080        BNE ?5
2090        LDA HMSB
2100        ORA BITPOS,X
2110        STA HMSB
2120        BNE ?6          ;BRANCH ALWAYS
2130 ;
2140 ?5     LDA HMSB
2150        AND BITAND,X
2160        STA HMSB
2170 ?6     .MND
2180 ;
2190        .MAC UNPACK     ;LOAD X--SRC.MSB.DST
2200        LDA ?2
2210        AND BITPOS,X
2220        BEQ ?4
2230        LDA #$01
2240 ?4     STA ?3+1
2250        LDA ?1
```

```
2260         STA ?3
2270         .MND
2280 ;
2290         .MAC PUNPCK
2300         LDA ?2
2310         AND BITPOS,X
2320         BEQ ?4
2330         LDA #$01
2340 ?4      STA ?3+1
2350         LDA ?1,X
2360         STA ?3
2370         .MND
2380 ;
2390         .MAC SMNPC        ;SRC,MSB,DST--
2400         LDA ?2
2410         AND BITPOS,X
2420         BEQ ?4
2430         LDA #$01
2440 ?4      STA ?3
2450         ROR ?3
2460         LDA ?1
2470         ROR A
2480         STA ?3
2490         .MND
2500 ;
2510         .MAC SMNPCX       ;INDEXED BY X
2520         LDA ?2
2530         AND BITPOS,X
2540         BEQ ?4
2550         LDA #$01
2560 ?4      STA ?3
2570         ROR ?3
2580         LDA ?1,X
2590         ROR A
2600         STA ?3
2610         .MND
2620 ;
2630 ;
2640 ;
2650 ;
2660 ;
2670         .MAC DBADC        ;A,+B,=C
2680         LDA ?1
2690         CLC
2700         ADC ?2
2710         STA ?3
2720         LDA ?1+1
2730         ADC ?2+1
2740         STA ?3+1
2750         .MND
2760 ;
2770         .MAC DBADCI       ;A,+B,=C
2780         LDA ?1
2790         CLC
2800         ADC ?2
2810         STA ?3
2820         LDA ?1+1
2830         ADC #$00
2840         STA ?3+1
```

```
2850          .MND
2860 ;
2870 ;
2880          .MAC DBSBC        ;A,-B,=C
2890          LDA ?1
2900          SEC
2910          SBC ?2
2920          STA ?3
2930          LDA ?1+1
2940          SBC ?2+1
2950          STA ?3+1
2960          .MND
2970 ;
2980          .MAC DBSBCI       ;A,-B,=C
2990          LDA ?1
3000          SEC
3010          SBC ?2
3020          STA ?3
3030          LDA ?1+1
3040          SBC #$00
3050          STA ?3+1
3060          .MND
3070 ;
3080 ;
3090          .MAC NIBLR        ;MOVE NIBBLE RIGHT
3100          LSR A
3110          LSR A
3120          LSR A
3130          LSR A
3140          .MND
3150 ;
3160 ;
3170          .MAC NIBLL        ;MOVE NIBBLE LEFT
3180          ASL A
3190          ASL A
3200          ASL A
3210          ASL A
3220          .MND
3230 ;
3240 ;
3250          .MAC ASL2         ;TWO BYTE ASL
3260          ASL ?1
3270          ROL ?1+1
3280          .MND
3290 ;
3300 ;
3310          .MAC LSR2         ;TWO BYTE LSR
3320          LSR ?1
3330          ROR ?1+1
3340          .MND
3350 ;
3360 ;
3370          .MAC BGT          ;BRAANCH IF >
3380          BEQ ?4
3390          BCS ?1
3400 ?4       .MND
3410 ;
3420 ;
3430          .MAC BLE          ;BRANCH IF <=
```

```
3440          BCC ?1
3450          BEQ ?1
3460          .MND
3470 ;
3480 ;
3490          .MAC DBNE         ;DEC BRANCH IF NOT 0
3500          DE?1
3510          BNE ?2
3520          .MND
3530 ;
3540 ;
3550          .MAC DJNE
3560          DE?1
3570          BEQ ?4
3580          JMP ?2
3590 ?4       .MND
3600 ;
3610 ;
3620 ;
3630 ;
3640 ;
3650          .MAC DBPL         ;DEC BRANCH +
3660          DE?1
3670          BPL ?2
3680          .MND
3690 ;
3700 ;
3710          .MAC JCC          ;JUMP IF CARRY CLR
3720          BCS ?4
3730          JMP ?1
3740 ?4       .MND
3750 ;
3760 ;
3770          .MAC JCS
3780          BCC ?4
3790          JMP ?1
3800 ?4       .MND
3810 ;
3820 ;
3830          .MAC JEQ
3840          BNE ?4
3850          JMP ?1
3860 ?4       .MND
3870 ;
3880 ;
3890          .MAC JGE
3900          BLT ?4
3910          JMP ?1
3920 ?4       .MND
3930 ;
3940 ;
3950          .MAC JGT
3960          BEQ ?4
3970          BCC ?4
3980          JMP ?1
3990 ?4       .MND
4000 ;
4010 ;
4020          .MAC JLE
```

```
4030          BEQ ?3
4040          BCS ?4
4050 ?3       JMP ?1
4060 ?4       .MND
4070 ;
4080 ;
4090          .MAC JLT
4100          BCS ?4
4110          JMP ?1
4120 ?4       .MND
4130 ;
4140 ;
4150          .MAC JMI
4160          BPL ?4
4170          JMP ?1
4180 ?4       .MND
4190 ;
4200 ;
4210          .MAC JNE
4220          BEQ ?4
4230          JMP ?1
4240 ?4       .MND
4250 ;
4260 ;
4270          .MAC JPL
4280          BMI ?4
4290          JMP ?1
4300 ?4       .MND
4310 ;
4320 ;
4330          .MAC LDMEM
4340          LDA ?1
4350          STA ?2
4360          .MND
4370 ;
4380 ;
4390          .MAC LDMEMX
4400          LDX ?1
4410          STX ?2
4420          .MND
4430 ;
4440 ;
4450          .MAC LDMEMY
4460          LDY ?1
4470          STY ?2
4480          .MND
4490 ;
4500 ;
4510          .MAC BANK
4520          LDA #$03
4530          STA $DD02
4540          SEC
4550          SBC #?1
4560          STA $DD00
4570          .MND
4580 ;
4590 ;
4600          .MAC MVMEM        ;SRC,DST,#PAGES
4610          ADRES ?1,SCRPT
```

```
4620        ADRES ?2,SCRDST
4630        LDY #$00
4640        LDX #$00
4650 ?5     LDA (SCRPT),Y
4660        STA (SCRDST),Y
4670        INY
4680        BNE ?5
4690        INC SCRPT+1
4700        INC SCRDST+1
4710        INX
4720        CPX #?3
4730        BNE ?5
4740        .MND
4750 ;
4760 ;
4770        .MAC MVCOL       ;COLOR
4780        ADRES $D800,SCRDST
4790        LDY #$00
4800        LDX #$00
4810 ?4     LDA #?1
4820 ?5     STA (SCRDST),Y
4830        INY
4840        BNE ?5
4850        INC SCRDST+1
4860        LDA SCRDST+1
4870        CMP #$DC
4880        BNE ?4
4890        .MND
4900 ;
4910 ;
4920        .MAC FILL
4930        ADRES ?1,SCRDST
4940        LDY #$00
4950        LDX #$00
4960 ?4     LDA #?2
4970 ?5     STA (SCRDST),Y
4980        INY
4990        BNE ?5
5000        INC SCRDST+1
5010        INX
5020        CPX #?3
5030        BNE ?4
5040        .MND
5050 ;
5060 ;
5070        .MAC FILBYT
5080        ADRES ?1,DST
5090        LDY #$00
5100        LDA #?2
5110 ?4     STA (DST),Y
5120        INY
5130        CPY #?3
5140        BNE ?4
5150        .MND
5160 ;
5170 ;
5180        .MAC ANOP
5190        .MND
5200 ;
```

```
5210 ;
5220        .MAC TEXT
5230        LDA YSCRL
5240        AND #$DF
5250        STA YSCRL
5260        .MND
5270 ;
5280 ;
5290        .MAC GRAPH
5300        LDA YSCRL
5310        ORA #$20
5320        STA YSCRL
5330        .MND
5340 ;
5350 ;
5360        .MAC MULTON
5370        LDA XSCRL
5380        ORA #$10
5390        STA XSCRL
5400        .MND
5410 ;
5420 ;
5430        .MAC MULTOF
5440        LDA XSCRL
5450        AND #$EF
5460        STA XSCRL
5470        .MND
5480 ;
5490 ;
5500        .MAC BLNK
5510        LDA YSCRL
5520        AND #$EF
5530        STA YSCRL
5540        .MND
5550 ;
5560 ;
5570        .MAC UNBLNK
5580        LDA YSCRL
5590        ORA #$10
5600        STA YSCRL
5610        .MND
5620 ;
5630 ;
5640        .MAC TXBAS
5650        LDA #>?1
5660        ASL A
5670        ASL A
5680        STA BUF1
5690        LDA VIDBAS
5700        AND #$0F
5710        ORA BUF1
5720        STA VIDBAS
5730        .MND
5740 ;
5750 ;
5760        .MAC GRABAS
5770        LDA #>?1
5780        LSR A
5790        LSR A
```

```
5800         STA BUF1
5810         LDA VIDBAS
5820         AND #$F0
5830         ORA BUF1
5840         STA VIDBAS
5850         .MND
5860 ;
5870 ;
5880 ;
5890 ;
5900 ;
5910         .MAC NOT          ;COMPLEMENTS ACC
5920         EOR #$FF
5930         .MND
5940 ;
5950         .MAC TPINC        ;THREE BYTE INC
5960         LDA ?1
5970         CLC
5980         ADC #?2
5990         STA ?1
6000         LDA #$00
6010         ADC ?1+1
6020         STA ?1+1
6030         LDA #$00
6040         ADC ?1+2
6050         STA ?1+2
6060         .MND
6070 ;
6080 ;
6090         .MAC QDINC        ;FOUR BYTE INC
6100         LDA ?1
6110         CLC
6120         ADC #?2
6130         STA ?1
6140         LDA #$00
6150         ADC ?1+1
6160         STA ?1+1
6170         LDA #$00
6180         ADC ?1+2
6190         STA ?1+2
6200         LDA #$00
6210         ADC ?1+3
6220         STA ?1+3
6230         .MND
6240 ;
6250 ;
6260         .MAC TPDEC        ;3 BYTE DEC
6270         LDA ?1
6280         SEC
6290         SBC #?2
6300         STA ?1
6310         LDA ?1+1
6320         SBC #$00
6330         STA ?1+1
6340         LDA ?1+2
6350         SBC #$00
6360         STA ?1+2
6370         .MND
6380 ;
```

```
6390 ;
6400          .MAC QDDEC        ;4 BYTE DEC
6410          LDA ?1
6420          SEC
6430          SBC #?2
6440          STA ?1
6450          LDA ?1+1
6460          SBC #$00
6470          STA ?1+1
6480          LDA ?1+2
6490          SBC #$00
6500          STA ?1+2
6510          LDA ?1+3
6520          SBC #$00
6530          STA ?1+3
6540          .MND
6550 ;
6560 ;
6570          .END
```

Listing C-2: The SYSDEF Source Code

```
1000 ;PUT"@0:SYSDEF"
1010 ;
1020 ;FOR THE COMMODORE-64
1030 ;SYSTEM DEFINITIONS
1040 ;SYSDEF
1050 ;
1060 ;
1070 JOY1   = $DC00
1080 JOY2   = $DC01
1090 DDRA   = $DC02
1100 DDRB   = $DC03
1110 ;
1120 ;VIC CHIP ADDRESSES
1130 ;
1140 SPR0X  = $D000
1150 SPR0Y  = $D001
1160 SPR1X  = $D002
1170 SPR1Y  = $D003
1180 SPR2X  = $D004
1190 SPR2Y  = $D005
1200 SPR3X  = $D006
1210 SPR3Y  = $D007
1220 SPR4X  = $D008
1230 SPR4Y  = $D009
1240 SPR5X  = $D00A
1250 SPR5Y  = $D00B
1260 SPR6X  = $D00C
1270 SPR6Y  = $D00D
1280 SPR7X  = $D00E
1290 SPR7Y  = $D00F
1300 XMSB   = $D010
1310 YSCRL  = $D011
1320 RASTER = $D012
1330 LPX    = $D013
1340 LPY    = $D014
1350 SPREN  = $D015
1360 XSCRL  = $D016
1370 SPRYSZ = $D017
1380 VIDBAS = $D018
1390 VIRQ   = $D019
1400 VIRQM  = $D01A
1410 SPRIOR = $D01B
1420 MLTSP  = $D01C
1430 SPRXSZ = $D01D
1440 SSCOL  = $D01E
1450 SBCOL  = $D01F
1460 ;
1470 BORCL  = $D020
1480 BCOL0  = $D021
1490 BCOL1  = $D022
1500 BCOL2  = $D023
1510 SPMCL0 = $D025
1520 SPMCL1 = $D026
1530 SPRCL0 = $D027
1540 SPRCL1 = $D028
1550 SPRCL2 = $D029
1560 SPRCL3 = $D02A
1570 SPRCL4 = $D02B
1580 SPRCL5 = $D02C
1590 SPRCL6 = $D02D
1600 SPRCL7 = $D02E
1610 V1FRLO = $D400 ;VOICE 1 FREQ LO
1620 V1FRHI = $D401
1630 V1PWLO = $D402 ;VOICE 1 PULSE WIDTH
1640 V1PWHI = $D403
1650 V1CORG = $D404
1660 V1ATDC = $D405
1670 V1SURL = $D406
1680 ;
1690 V2FRLO = $D407
1700 V2FRHI = $D408
1710 V2PWLO = $D409
```

```
1720 V2PWHI = $D40A
1730 V2CORG = $D40B
1740 V2ATDC = $D40C
1750 V2SURL = $D40D
1760 ;
1770 V3FRLO = $D40E
1780 V3FRHI = $D40F
1790 V3PWLO = $D410
1800 V3PWHI = $D411
1810 V3CORG = $D412
1820 V3ATDC = $D413
1830 V3SURL = $D414
1840 ;
1850 FLCNLO = $D415
1860 FLCNHI = $D416
1870 RESFLT = $D417
1880 MODVOL = $D418
1890 ;
1900 SPOTX  = $D419
1910 SPOTY  = $D41A
1920 RANDOM = $D41B
1930 ENVLOP = $D41C
1940 ;
1950 ;
1960 ORGVAL = $8000
1970 ;
1980 ;
1990 ;COLORS
2000 ;
2010 BLACK  = 0
2020 WHITE  = $1
2030 RED    = $2
2040 CYAN   = $3
2050 PURPL  = $4
2060 GREEN  = $5
2070 BLUE   = $6
2080 YELOW  = $7
2090 ORNGE  = $8
2100 BROWN  = $9
2110 LTRED  = $A
2120 GRAY1  = $B
2130 GRAY2  = $C
2140 LTGRN  = $D
2150 LTBLU  = $E
2160 GRAY3  = $F
2170 OSYS   = $EA31
2180 ;
2190 ;
2200 CINV   = $0314        ;INT VECTOR
2210 NMINV  = $0318        ;NMI VECTOR
2220 PRTDIR = $00          ;PORT DIR
2230 PORT   = $01          ;I/O PORT
2240 CIANT1 = $DC0D
2250 CIANT2 = $DD0D
2260 SPRPT1 = $5FF8
2270 SPRPT2 = $5FF9
2280 SPRPT3 = $5FFA
2290 SPRPT4 = $5FFB
2300 SPRPT5 = $5FFC
2310 SPRPT6 = $5FFD
2320 SPRPT7 = $5FFE
2330 SPRPT8 = $5FFF
2340 TXSCRN = $0400
2350 ;
2360 ;
2370         .END
```

Listing C-3: The INTER-DEMO Program, Source Code for the INT DEMO.O Program

```
1000 ;PUT"@0:INTER-DEMO"
1010 ;LOAD"ASM",8
1020 ;
1030 ;INTERRUPT DEMO
1040 ;
1050 ;TO START THE DEMO FROM BASIC
1060 ;TYPE 'SYS 4096'
1070 ;
1080 ;CREATED 7/17/84
1090 ;(C) COPYRIGHT 1984, STEVEN BRESS
1100 ;
1110 ;LATEST ADDITIONS 11/15/84
1120 ;03:30 PM
1130 ;
1140         .OPT NOLIST,NOSYM
1150 ;
1160         .PAGE 'MACRO LIBRARY'
1170         .LIB MACLIB
1180         .PAGE 'SYSTEM DEFINITIONS'
1190         .LIB SYSDEF
1200 ;
1210         .OPT LIST
```

```
1220 *       = $1000
1230         JMP TOP
1240 ;
1250         .PAGE 'INTERRUPT DEMO CODE'
1260 ;
1270 ;
1280 RET     RTI
1290 TOP     SEI
1300 ;
1310         KILL              ;SHUT OFF OPERATING SYSTEM
1320         ADRES INT0,CINV   ;CINV POINTS TO INT0
1330         RAST #$96         ;INTERRUPT AT MID SCREEN
1340         CLI               ;CLEAR INTERRUPT DISABLE
1350 TWID    JMP TWID          ;MAIN PROGRAM DOES NOTHING
1360 ;
1370 ;
1380 INT0    LDMEM #BLUE,BCOL0 ;CHANGE BACKGROUND COLOR
1390         RAST #$FB         ;NEXT INTERRUPT AT BOTTOM
1400         ADRES INT1,CINV   ;POINT TO NEXT INTERRUPT
1410         IPULL             ;RETURN
1420 ;
1430 ;
1440 INT1    LDMEM #GREEN,BCOL0 ;CHANGE COLOR
1450         RAST #$96         ;NEXT INTERRUPT AT MID SCREEN
1460         ADRES INT0,CINV   ;POINT TO FIRST INTERRUPT
1470         IPULL             ;RETURN
1480 ;
1490         NOP
1500         .END
```

Listing C-4: The INT DEMO.O Program

```
.:1000 4C 04 10 40 78 78 A9 06
.:1008 85 01 A9 00 8D 0D DC 8D
.:1010 0D DD AD 0D DC AD 0D DD
.:1018 A9 00 8D 02 DC 8D 03 DC
.:1020 8D 0E DC 8D 0F DC 8D 0E
.:1028 DD 8D 0F DD A9 01 8D 1A
.:1030 D0 A9 FF 8D 19 D0 A2 02
.:1038 A9 00 95 00 E8 D0 FB A2
.:1040 FF 9A A9 03 8D 18 03 A9
.:1048 10 8D 19 03 A9 67 8D 14
.:1050 03 A9 10 8D 15 03 A9 96
.:1058 8D 12 D0 AD 11 D0 29 7F
.:1060 8D 11 D0 58 4C 64 10 A9
.:1068 06 8D 21 D0 A9 FB 8D 12
.:1070 D0 AD 11 D0 29 7F 8D 11
.:1078 D0 A9 8E 8D 14 03 A9 10
.:1080 8D 15 03 A9 FF 8D 19 D0
.:1088 68 A8 68 AA 68 40 A9 05
.:1090 8D 21 D0 A9 96 8D 12 D0
.:1098 AD 11 D0 29 7F 8D 11 D0
.:10A0 A9 67 8D 14 03 A9 10 8D
.:10A8 15 03 A9 FF 8D 19 D0 68
.:10B0 A8 68 AA 68 40 EA 00 00
```

Listing C-5: The SOUND Program, Source Code for the SOUND DEMO Program

```
1000 ;PUT"@0:SOUND"
1010 ;LOAD"ASM",8
1020 ;
1030 ;SOUND EFFECTS
1040 ;
1050 ;CREATED 6/30/84
1060 ;LATEST ADDITIONS 7/02/84
1070 ;(C) COPYRIGHT 1984, STEVEN
     BRESS
1080 ;
1090         .OPT LIST,NOSYM,NOGEN
1100         .LIB MACLIB
1110         .LIB SYSDEF
1120         .LIB COMMON
1130         .LIB SNDDEF
1140 ;
1150 *       = $1000
1160 ;
1170         JMP TOP
1180 ;
```

```
1190          .LIB DATA
1200 ;
1210 ;
1220 TOP      SEI
1230          KILL
1240          ADRES INT0,CINV
1250          RAST #$FB
1260          MVCOL WHITE
1270          FILL $0400,$20,$04
1280          GRABAS $1800
1290 ;
1300 ;
1310          CLI
1320 START    ANOP
1330          LDMEM #$10,WAVTYP
1340 ;
1350 ;MIDDLE FREQUENCY BASE LINE
1360 ;TRIANGLE WAVE
1370 ;
1380          ADRES MTRI,SRC
1390          JSR PRINT
1400 ;
1410          LDMEM #$00,S3PWLO
1420          STA V3PWLO
1430          LDMEM #$08,S3PWHI
1440          STA V3PWHI
1450          LDMEM #$0F,MMOD
1460          STA MODVOL
1470          LDMEM #$00,RFIL
1480          STA RESFLT
1490          LDMEM #$00,FLCNLO
1500          STA FLCNHI
1510 ;
1520          LDMEM #$77,S3ATDC
1530          STA V3ATDC
1540          LDMEM #$77,S3SURL
1550          STA V3SURL
1560 ;
1570 NXTWAV   LDMEM WAVTYP,S3CORG
1580          STA V3CORG
1590          ADRES TUNES,NOTPT3
1600          LDY #$00
1610          LDA (NOTPT3),Y
1620          STA S3FRLO
1630          STA V3FRLO
1640          INY
1650          LDA (NOTPT3),Y
1660          STA S3FRHI
1670          STA V3FRHI
1680          DBINC NOTPT3,$02
1690 ;
1700          ADRES TUNTM,NOTTM3
1710          LDY #$00
1720          LDA (NOTTM3),Y
1730          STA SNDTM3
1740          DBINC NOTTM3,1
1750          LDA WAVTYP
1760          ORA #$01
1770          STA S3CORG
1780          STA V3CORG
1790 ;
1800          LDMEM #$00,ENABLE
1810 ;
1820 TWID1    LDA ENABLE
1830          BEQ TWID1
1840          ASL WAVTYP
1850          BEQ WVDN
1860          LDA WAVTYP
1870          CMP #$20
1880          BNE NXMSG1
1890          ADRES MSAW,SRC
1900          JSR PRINT
1910 NXMSG1   CMP #$40
1920          BNE NXMSG2
1930          ADRES MSQU,SRC
1940          JSR PRINT
1950 NXMSG2   CMP #$80
1960          BNE NXMSG3
1970          ADRES MNOI,SRC
1980          JSR PRINT
1990 ;
2000 NXMSG3   JMP NXTWAV
2010 ;
2020 ;
2030 WVDN     ANOP
2040 ;
2050 ;CHANGING PULSEWIDTH
2060 ;
2070          SEI
2080          LDMEM #$40,WAVTYP
2090          LDMEM #$00,V3ATDC
2100          STA S3ATDC
2110          LDMEM #$FF,V3SURL
2120          STA S3SURL
2130          ADRES MCPW,SRC
2140          JSR PRINT
2150          ADRES INT1,CINV
2160          CLI
2170          ADRES AN4,S3FRLO
2180          LDMEM S3FRLO,V3FRLO
2190          LDMEM S3FRHI,V3FRHI
2200 PWCH1    DBDEC S3PWLO,$08
2210          LDMEM S3PWLO,V3PWLO
2220          LDMEM S3PWHI,V3PWHI
2230 ;
2240 ;
2250          LDMEM #$00,ENABLE
2260 TWID2    LDA ENABLE
2270          BEQ TWID2
2280          LDA S3PWLO
2290          ORA S3PWHI
2300          BNE PWCH1
2310 ;
2320          LDMEM #$00,S3CORG
2330          STA V3CORG
2340 ;
2350 ;
2360 ;CHANGING OCTAVES
```

```
2370 ;
2380          ADRES MOCT,SRC
2390          JSR PRINT
2400 ;
2410          LDMEM #$20,WAVTYP
2420          ADRES OCTAVE,NOTPT3
2430          ADRES OCTTM,NOTTM3
2440 ;
2450          ADRES INT0,CINV
2460 ;
2470          LDMEM #$20,S3CORG
2480          STA V3CORG
2490 ;
2500          LDY #$00
2510          LDA (NOTPT3),Y
2520          STA V3FRLO
2530          STA S3FRLO
2540 ;
2550          LDA (NOTTM3),Y
2560          STA SNDTM3
2570 ;
2580          INY
2590          LDA (NOTPT3),Y
2600          STA V3FRHI
2610          STA S3FRHI
2620 ;
2630          DBINC NOTPT3,2
2640          DBINC NOTTM3,1
2650 ;
2660          LDMEM #$77,V3ATDC
2670          STA S3ATDC
2680          STA V3SURL
2690          STA S3SURL
2700 ;
2710 ;
2720          LDMEM #$21,V3CORG
2730          STA S3CORG
2740 ;
2750          LDMEM #$00,ENABLE
2760 TWID3    LDA ENABLE
2770          BEQ TWID3
2780 ;
2790 ;
2800 ;CHANGING ATTACK TIME
2810 ;
2820          ADRES MATT,SRC
2830          JSR PRINT
2840 ;
2850          SEI
2860          ADRES INT1,CINV
2870          CLI
2880 ;
2890          LDMEM #$01,WAVTYP
2900 ;
2910          LDMEM #$05,S3ATDC
2920          STA V3ATDC
2930          LDMEM #$03,V3SURL
2940          STA S3SURL
2950          ADRES BN4,S3FRLO
2960          LDMEM S3FRLO,V3FRLO
2970          LDMEM S3FRHI,V3FRHI
2980 ;
2990 CHAT     LDMEM #$21,S3CORG
3000          STA V3CORG
3010          LDA S3ATDC
3020          NIBLR
3030          TAX
3040          LDA ATLK,X
3050          STA SNDTM3
3060 ;
3070 ;
3080          LDMEM #$00,TIMOUT
3090 TWID4    LDA TIMOUT
3100          BEQ TWID4
3110 ;
3120          LDMEM #$20,S3CORG
3130          STA V3CORG
3140          LDA S3ATDC
3150          CLC
3160          ADC #$10
3170          STA S3ATDC
3180          STA V3ATDC
3190          AND #$F0
3200          BNE CHAT
3210 ;
3220 ;
3230 ;CHANGE DECAY NEXT
3240 ;
3250          ADRES MDEC,SRC
3260          JSR PRINT
3270 ;
3280          LDMEM #$01,WAVTYP
3290          LDMEM #$50,S3ATDC
3300          STA V3ATDC
3310          LDMEM #$03,S3SURL
3320          STA V3SURL
3330          ADRES CN4,S3FRLO
3340          LDMEM S3FRLO,V3FRLO
3350          LDMEM S3FRHI,V3FRHI
3360 ;
3370 CHDC     LDMEM #$21,S3CORG
3380          STA V3CORG
3390          LDA S3ATDC
3400          AND #$0F
3410          TAX
3420          LDA DCLK,X
3430          STA SNDTM3
3440 ;
3450          LDMEM #$00,TIMOUT
3460 TWID5    LDA TIMOUT
3470          BEQ TWID5
3480          LDMEM #$20,S3CORG
3490          STA V3CORG
3500          LDA S3ATDC
3510          CLC
3520          ADC #$01
3530          STA S3ATDC
3540          STA V3ATDC
```

```
3550         AND #$0F
3560         BNE CHDC
3570 ;
3580 ;
3590 ;CHANGE SUSTAIN
3600 ;
3610         ADRES MSUS,SRC
3620         JSR PRINT
3630 ;
3640         LDMEM #$20,V3ATDC
3650         STA S3ATDC
3660         LDMEM #$00,S3SURL
3670         STA V3SURL
3680         ADRES DN4,S3FRLO
3690         LDMEM S3FRLO,V3FRLO
3700         LDMEM S3FRHI,V3FRHI
3710 ;
3720 CHSU    LDMEM #$21,S3CORG
3730         STA V3CORG
3740         LDMEM #$01,SNDTM3
3750         LDMEM #$00,TIMOUT
3760 ;
3770 TWID6   LDA TIMOUT
3780         BEQ TWID6
3790         LDMEM #$20,S3CORG
3800         STA V3CORG
3810         LDA S3SURL
3820         CLC
3830         ADC #$10
3840         STA S3SURL
3850         STA V3SURL
3860         AND #$F0
3870         BNE CHSU
3880 ;
3890 ;CHANGE RELEASE
3900 ;
3910         ADRES MREL,SRC
3920         JSR PRINT
3930 ;
3940         LDMEM #$F0,S3SURL
3950         STA V3SURL
3960         ADRES EN4,S3FRLO
3970         LDMEM S3FRLO,V3FRLO
3980         LDMEM S3FRHI,V3FRHI
3990 ;
4000 CHRE    LDMEM #$21,S3CORG
4010         STA V3CORG
4020         LDMEM #$01,SNDTM3
4030 ;
4040 ;
4050         LDMEM #$00,TIMOUT
4060 TWID7   LDA TIMOUT
4070         BEQ TWID7
4080 ;
4090         LDMEM #$20,S3CORG
4100         STA V3CORG
4110         LDA S3SURL
4120         AND #$0F
4130         TAX
4140         LDA DCLK,X
4150         STA SNDTM3
4160 ;
4170         LDMEM #$00,TIMOUT
4180 TWID8   LDA TIMOUT
4190         BEQ TWID8
4200 ;
4210         LDA S3SURL
4220         CLC
4230         ADC #$01
4240         STA S3SURL
4250         STA V3SURL
4260         AND #$0F
4270         BNE CHRE
4280 ;
4290 ;
4300 ;CHANGE LO PASS FILTER
4310 ;
4320         ADRES MFLO,SRC
4330         JSR PRINT
4340 ;
4350         LDMEM #$1F,MMOD
4360         STA MODVOL
4370         LDMEM #$06,RESFLT
4380         STA RFIL
4390 ;
4400         ADRES $0000,S3FRLO
4410         LDMEM S3FRLO,V3FRLO
4420         LDMEM S3FRHI,V3FRHI
4430         ADRES $0000,S2FRLO
4440         LDMEM S2FRLO,V2FRLO
4450         LDMEM S2FRHI,V2FRHI
4460 ;
4470         LDMEM #$00,FILLO
4480         STA FLCNLO
4490         LDMEM #$08,FILHI
4500         STA FLCNHI
4510 ;
4520         LDMEM #$00,S3ATDC
4530         STA V3ATDC
4540         STA V2ATDC
4550         STA S2ATDC
4560         LDMEM #$F0,S3SURL
4570         STA V3SURL
4580         STA S2SURL
4590         STA V2SURL
4600 ;
4610 ;
4620         LDMEM #$21,S3CORG
4630         STA V3CORG
4640         STA S2CORG
4650         STA V2CORG
4660 CHLFL   LDMEM #$00,ENABLE
4670 TWID9   LDA ENABLE
4680         BEQ TWID9
4690 ;
4700         DBINC S3FRLO,$20
4710         DBDEC S2FRLO,$20
4720         LDMEM S2FRLO,V2FRLO
```

```
4730          LDMEM S2FRHI,V2FRHI
4740          LDMEM S3FRLO,V3FRLO
4750          LDMEM S3FRHI,V3FRHI
4760          LDA S3FRLO
4770          ORA S3FRHI
4780          BNE CHLFL
4790 ;
4800 ;CHANGE HIGH-PASS FILTER
4810 ;
4820          ADRES MFHI,SRC
4830          JSR PRINT
4840 ;
4850          ADRES $0000,S3FRLO
4860          ADRES $0000,S2FRLO
4870          LDMEM S3FRLO,V3FRLO
4880          LDMEM S2FRLO,V2FRLO
4890          LDMEM S3FRHI,V3FRHI
4900          LDMEM S2FRHI,V2FRHI
4910          LDMEM #$4F,MMOD
4920          STA MODVOL
4930 ;
4940 CHHFL    LDMEM #$00,ENABLE
4950 TWIDA    LDA ENABLE
4960          BEQ TWIDA
4970 ;
4980          DBINC S3FRLO,$20
4990          DBDEC S2FRLO,$20
5000          LDMEM S2FRLO,V2FRLO
5010          LDMEM S2FRHI,V2FRHI
5020          LDMEM S3FRLO,V3FRLO
5030          LDMEM S3FRHI,V3FRHI
5040          LDA S3FRLO
5050          ORA S3FRHI
5060          BNE CHHFL
5070 ;
5080 ;CHANGE BAND-PASS FILTER
5090 ;
5100          ADRES MFBA,SRC
5110          JSR PRINT
5120 ;
5130          ADRES $0000,S3FRLO
5140          ADRES $0000,S3FRLO
5150          LDMEM S3FRLO,V3FRLO
5160          LDMEM S2FRLO,V2FRLO
5170          LDMEM S3FRHI,V3FRHI
5180          LDMEM S2FRHI,V2FRHI
5190          LDMEM #$2F,MMOD
5200          STA MODVOL
5210 ;
5220 CHBFL    LDMEM #$00,ENABLE
5230 TWIDB    LDA ENABLE
5240          BEQ TWIDB
5250 ;
5260          DBINC S3FRLO,$20
5270          DBDEC S2FRLO,$20
5280          LDMEM S2FRLO,V2FRLO
5290          LDMEM S2FRHI,V2FRHI
5300          LDMEM S3FRLO,V3FRLO
5310          LDMEM S3FRHI,V3FRHI
5320          LDA S3FRLO
5330          ORA S3FRHI
5340          BNE CHBFL
5350          STA S3CORG
5360          STA V3CORG
5370          STA S2CORG
5380          STA V2CORG
5390 ;
5400 ;CHANGING RESONANCE
5410 ;
5420          ADRES MRES,SRC
5430          JSR PRINT
5440 ;
5450          LDMEM #$81,S3CORG
5460          STA V3CORG
5470          LDMEM #$80,FLCNHI
5480          STA FILHI
5490          ADRES GN4,S3FRLO
5500          LDMEM S3FRLO,V3FRLO
5510          LDMEM S3FRHI,V3FRHI
5520          LDMEM #$1F,MMOD
5530          STA MODVOL
5540 ;
5550 CHRES    LDMEM #$01,SNDTM3
5560          LDMEM #$00,TIMOUT
5570 TWIDC    LDA TIMOUT
5580          BEQ TWIDC
5590 ;
5600          LDA RFIL
5610          CLC
5620          ADC #$10
5630          STA RFIL
5640          STA RESFLT
5650          AND #$F0
5660          BNE CHRES
5670 ;
5680 ;RING MODULATION ON
5690 ;
5700          ADRES MRMO,SRC
5710          JSR PRINT
5720 ;
5730          LDMEM #$77,S1ATDC
5740          STA V1ATDC
5750          STA S1SURL
5760          STA V1SURL
5770          STA S3ATDC
5780          STA V3ATDC
5790          STA S3SURL
5800          STA V3SURL
5810          STA S2SURL
5820          STA V2SURL
5830          STA S2ATDC
5840          STA V2ATDC
5850 ;
5860          ADRES DS0,S3FRLO
5870          LDMEM S3FRLO,V3FRLO
5880          LDMEM S3FRHI,V3FRHI
5890          LDA MMOD
5900          ORA #$80
```

```
5910         STA MODVOL
5920         LDMEM #$20,S3CORG
5930         STA V3CORG
5940         LDMEM #$14,S1CORG
5950         ADRES INT0,CINV
5960         ADRES MODLK,NOTPT1
5970         ADRES MODTM,NOTTM1
5980         LDY #$00
5990         LDA (NOTPT1),Y
6000         STA S1FRLO
6010         STA V1FRLO
6020         LDA (NOTTM1),Y
6030         STA SNDTM1
6040         INY
6050         LDA (NOTPT1),Y
6060         STA S1FRHI
6070         STA V1FRHI
6080         DBINC NOTTM1,$01
6090         DBINC NOTPT1,$02
6100 ;
6110         LDMEM #$15,S1CORG
6120         STA V1CORG
6130 ;
6140         LDMEM #$00,ENABLE
6150 TWIDD   LDA ENABLE
6160         BEQ TWIDD
6170 ;
6180 ;CHANGING SYNCHRONIZATION
6190 ;
6200         ADRES MSYN,SRC
6210         JSR PRINT
6220 ;
6230         LDMEM #$22,S1CORG
6240         STA V1CORG
6250         ADRES MODLK,NOTPT1
6260         ADRES MODTM,NOTTM1
6270         LDY #$00
6280         LDA (NOTPT1),Y
6290         STA S1FRLO
6300         STA V1FRLO
6310         LDA (NOTTM1),Y
6320         STA SNDTM1
6330         INY
6340         LDA (NOTPT1),Y
6350         STA S1FRHI
6360         STA V1FRHI
6370         DBINC NOTTM1,$01
6380         DBINC NOTPT1,$02
6390 ;
6400         LDMEM #$23,S1CORG
6410         STA V1CORG
6420         LDMEM #$00,ENABLE
6430 TWIDE   LDA ENABLE
6440         BEQ TWIDE
6450 ;
6460 ;
6470         JMP TOP
6480 ;
6490 ;
6500 ;
6510 PRINT   ANOP
6520         FILBYT $0590,$20,$28
6530         ADRES $0590,DST
6540         LDY #$00
6550 PRTLP   LDA (SRC),Y
6560         BEQ PRTND
6570         STA (DST),Y
6580         INY
6590         JMP PRTLP
6600 PRTND   RTS
6610 ;
6620 ;
6630 INT0    SEI
6640         INC SCREEN
6650         INC RANSEC
6660         LDA RANSEC
6670         EOR #$3C
6680         BNE SYNC1
6690 ;
6700         STA RANSEC
6710         INC SECOND
6720 ;
6730 SYNC1   ANOP
6740 ;
6750         LDA SNDTM1
6760         BEQ DECTM1
6770         DEC SNDTM1
6780         BNE DECTM1
6790         LDA S1CORG
6800         AND #$FE
6810         STA S1CORG
6820         STA V1CORG
6830         LDY #$00
6840         LDA (NOTPT1),Y
6850         BNE MMUS1
6860         LDMEM #$01,ENABLE
6870         JMP DECTM1
6880 ;
6890 MMUS1   STA S1FRLO
6900         STA V1FRLO
6910         LDA (NOTTM1),Y
6920         STA SNDTM1
6930         INY
6940         LDA (NOTPT1),Y
6950         STA S1FRHI
6960         STA V1FRHI
6970         DBINC NOTTM1,$01
6980         DBINC NOTPT1,$02
6990         LDA S1CORG
7000         ORA #$01
7010         STA S1CORG
7020         STA V1CORG
7030 DECTM1  ANOP
7040 ;
7050         LDA SNDTM3
7060         BEQ DECTM3
7070         DEC SNDTM3
7080         BNE DECTM3
```

```
7090          LDMEM #$10,S3CORG
7100          STA V3CORG
7110          LDY #$00
7120          LDA (NOTPT3),Y
7130          BNE MMUS3
7140          LDMEM #$01,ENABLE
7150          JMP DECTM3
7160 ;
7170 MMUS3    STA V3FRLO
7180          STA S3FRLO
7190          INY
7200          LDA (NOTPT3),Y
7210          STA V3FRHI
7220          STA S3FRHI
7230          DBINC NOTPT3,$02
7240 ;
7250          LDY #$00
7260          LDA (NOTTM3),Y
7270          STA SNDTM3
7280          DBINC NOTTM3,1
7290 ;
7300          LDA WAVTYP
7310          ORA #$01
7320          STA S3CORG
7330          STA V3CORG
7340 ;
7350 DECTM3 ANOP
7360 ;
7370 ;LDMEM #$01,ENABLE
7380 ;
7390          RAST #$FB
7400          PULL
7410          NOP
7420 ;
7430 ;
7440 INT1     SEI
7450          INC SCREEN
7460          INC RANSEC
7470          LDA RANSEC
7480          EOR #$3C
7490          BNE SYNC2
7500 ;
7510          STA RANSEC
7520          INC SECOND
7530 ;
7540          LDA SNDTM3
7550          BEQ SYNC2
7560          DEC SNDTM3
7570          BNE SYNC2
7580          LDMEM #$01,TIMOUT
7590 ;
7600 SYNC2    LDMEM #$01,ENABLE
7610 ;
7620          LDA WAVTYP
7630          CMP #$01
7640          BEQ LEAVE
7650          LDMEM #$40,V3CORG
7660          LDMEM #$41,S3CORG
7670 ;
7680 ;
7690 LEAVE    RAST #$FB
7700          PULL
7710          NOP
7720 ;
7730 ;
7740          .END
```

Listing C-6: The SOUND DEMO Program

```
B*
   PC  SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:1000 4C 4B 12 40 28 43 29 20
.:1008 43 4F 50 59 52 49 47 48
.:1010 54 20 31 39 38 34 2C 20
.:1018 53 54 45 56 45 4E 20 42
.:1020 52 45 53 53 54 52 49 41
.:1028 4E 47 4C 45 20 57 41 56
.:1030 45 46 4F 52 4D 00 53 41
.:1038 57 54 4F 4F 54 48 20 57
.:1040 41 56 45 46 4F 52 4D 00
.:1048 53 51 55 41 52 45 20 57
.:1050 41 56 45 46 4F 52 4D 00
.:1058 4E 4F 49 53 45 20 57 41
.:1060 56 45 46 4F 52 4D 00 43
.:1068 48 41 4E 47 49 4E 47 20
.:1070 50 55 4C 53 45 57 49 44
.:1078 54 48 00 43 48 41 4E 47
.:1080 49 4E 47 20 4F 43 54 41
.:1088 56 45 53 00 43 48 41 4E
.:1090 47 49 4E 47 20 41 54 54
.:1098 41 43 4B 20 54 49 4D 45
.:10A0 00 43 48 41 4E 47 49 4E
.:10A8 47 20 44 45 43 41 59 20
.:10B0 54 49 4D 45 00 43 48 41
.:10B8 4E 47 49 4E 47 20 53 55
.:10C0 53 54 41 49 4E 20 4C 45
.:10C8 56 45 4C 00 43 48 41 4E
.:10D0 47 49 4E 47 20 52 45 4C
.:10D8 45 41 53 45 20 54 49 4D
.:10E0 45 00 43 48 41 4E 47 49
.:10E8 4E 47 20 4C 4F 2D 50 41
.:10F0 53 53 20 46 49 4C 54 45
.:10F8 52 00 43 48 41 4E 47 49
.:1100 4E 47 20 48 49 2D 50 41
.:1108 53 53 20 46 49 4C 54 45
.:1110 52 00 43 48 41 4E 47 49
.:1118 4E 47 20 42 41 4E 44 2D
.:1120 50 41 53 53 20 46 49 4C
.:1128 54 45 52 00 43 48 41 4E
```

```
.:1130 47 49 4E 47 20 52 45 53
.:1138 4F 4E 41 4E 43 45 00 52
.:1140 49 4E 47 20 4D 4F 44 55
.:1148 4C 41 54 49 4F 4E 20 4F
.:1150 4E 00 43 48 41 4E 47 49
.:1158 4E 47 20 4D 4F 44 55 4C
.:1160 41 54 4F 52 00 53 59 4E
.:1168 43 48 52 4F 4E 49 5A 41
.:1170 54 49 4F 4E 20 4F 4E 00
.:1178 43 48 41 4E 47 49 4E 47
.:1180 20 53 59 4E 43 20 46 52
.:1188 45 51 55 45 4E 43 49 45
.:1190 53 00 0C 01 18 02 30 04
.:1198 61 08 C3 10 87 21 0F 43
.:11A0 1E 86 00 00 3C 3C 3C 3C
.:11A8 3C 3C 3C 3C 00 0C 07 E9
.:11B0 07 61 08 8F 0C 30 0B 8F
.:11B8 0A 68 09 61 08 00 00 0F
.:11C0 0F 0F 0F 0F 0F 0F 0F 00
.:11C8 01 01 01 01 01 01 01 01
.:11D0 01 02 02 02 02 04 06 09
.:11D8 01 01 01 01 01 01 01 01
.:11E0 01 02 02 03 04 0A 10 1A
.:11E8 0C 01 51 01 91 01 C3 01
.:11F0 18 02 A3 02 23 03 86 03
.:11F8 30 04 47 05 47 06 0C 07
.:1200 61 08 8F 0A 8F 0C 18 0E
.:1208 C3 10 1F 15 32 19 31 1C
.:1210 87 21 3E 2A 3C 32 63 38
.:1218 0F 43 7D 54 79 64 C7 70
.:1220 1E 86 FA A8 F3 C8 8F E1
.:1228 00 00 3C 3C 3C 3C 3C 3C
.:1230 3C 3C 3C 3C 3C 3C 3C 3C
.:1238 3C 3C 3C 3C 3C 3C 3C 3C
.:1240 3C 3C 3C 3C 3C 3C 3C 3C
.:1248 3C 3C 00 78 78 A9 06 85
.:1250 01 A9 00 8D 0D DC 8D 0D
.:1258 DD AD 0D DC AD 0D DD A9
.:1260 00 8D 02 DC 8D 03 DC 8D
.:1268 0E DC 8D 0F DC 8D 0E DD
.:1270 8D 0F DD A9 01 8D 1A D0
.:1278 A9 FF 8D 19 D0 A2 02 A9
.:1280 00 95 00 E8 D0 FB A2 FF
.:1288 9A A9 03 8D 18 03 A9 10
.:1290 8D 19 03 A9 31 8D 14 03
.:1298 A9 19 8D 15 03 A9 FB 8D
.:12A0 12 D0 AD 11 D0 29 7F 8D
.:12A8 11 D0 A9 00 85 20 A9 D8
.:12B0 85 21 A0 00 A2 00 A9 01
.:12B8 91 20 C8 D0 FB E6 21 A5
.:12C0 21 C9 DC D0 F1 A9 00 85
.:12C8 20 A9 04 85 21 A0 00 A2
.:12D0 00 A9 20 91 20 C8 D0 FB
.:12D8 E6 21 E8 E0 04 D0 F2 A9
.:12E0 18 4A 4A 85 45 AD 18 D0
.:12E8 29 F0 05 45 8D 18 D0 58
.:12F0 A9 10 85 5D A9 24 85 22
.:12F8 A9 10 85 23 20 09 19 A9
.:1300 00 85 0F 8D 10 D4 A9 08
.:1308 85 10 8D 11 D4 A9 0F 85
.:1310 19 8D 18 D4 A9 00 85 1A
.:1318 8D 17 D4 A9 00 8D 15 D4
.:1320 8D 16 D4 A9 77 85 13 8D
.:1328 13 D4 A9 77 85 16 8D 14
.:1330 D4 A5 5D 85 04 8D 12 D4
.:1338 A9 AD 85 2A A9 11 85 2B
.:1340 A0 00 B1 2A 85 09 8D 0E
.:1348 D4 C8 B1 2A 85 0A 8D 0F
.:1350 D4 A5 2A 18 69 02 85 2A
.:1358 A9 00 65 2B 85 2B A9 BF
.:1360 85 30 A9 11 85 31 A0 00
.:1368 B1 30 85 1D A5 30 18 69
.:1370 01 85 30 A9 00 65 31 85
.:1378 31 A5 5D 09 01 85 04 8D
.:1380 12 D4 A9 00 85 3B A5 3B
.:1388 F0 FC 06 5D F0 32 A5 5D
.:1390 C9 20 D0 0B A9 36 85 22
.:1398 A9 10 85 23 20 09 19 C9
.:13A0 40 D0 0B A9 48 85 22 A9
.:13A8 10 85 23 20 09 19 C9 80
.:13B0 D0 0B A9 58 85 22 A9 10
.:13B8 85 23 20 09 19 4C 31 13
.:13C0 78 A9 40 85 5D A9 00 8D
.:13C8 13 D4 85 13 A9 FF 8D 14
.:13D0 D4 85 16 A9 67 85 22 A9
.:13D8 10 85 23 20 09 19 A9 FD
.:13E0 8D 14 03 A9 19 8D 15 03
.:13E8 58 A9 31 85 09 A9 1C 85
.:13F0 0A A5 09 8D 0E D4 A5 0A
.:13F8 8D 0F D4 A5 0F 38 E9 08
.:1400 85 0F A5 10 E9 00 85 10
.:1408 A5 0F 8D 10 D4 A5 10 8D
.:1410 11 D4 A9 00 85 3B A5 3B
.:1418 F0 FC A5 0F 05 10 D0 DB
.:1420 A9 00 85 04 8D 12 D4 A9
.:1428 7B 85 22 A9 10 85 23 20
.:1430 09 19 A9 20 85 5D A9 92
.:1438 85 2A A9 11 85 2B A9 A4
.:1440 85 30 A9 11 85 31 A9 31
.:1448 8D 14 03 A9 19 8D 15 03
.:1450 A9 20 85 04 8D 12 D4 A0
.:1458 00 B1 2A 8D 0E D4 85 09
.:1460 B1 30 85 1D C8 B1 2A 8D
.:1468 0F D4 85 0A A5 2A 18 69
.:1470 02 85 2A A9 00 65 2B 85
.:1478 2B A5 30 18 69 01 85 30
.:1480 A9 00 65 31 85 31 A9 77
.:1488 8D 13 D4 85 13 8D 14 D4
.:1490 85 16 A9 21 8D 12 D4 85
.:1498 04 A9 00 85 3B A5 3B F0
.:14A0 FC A9 8C 85 22 A9 10 85
.:14A8 23 20 09 19 78 A9 FD 8D
.:14B0 14 03 A9 19 8D 15 03 58
.:14B8 A9 01 85 5D A9 05 85 13
.:14C0 8D 13 D4 A9 03 8D 14 D4
.:14C8 85 16 A9 A5 85 09 A9 1F
.:14D0 85 0A A5 09 8D 0E D4 A5
.:14D8 0A 8D 0F D4 A9 21 85 04
```

```
.:14E0 8D 12 D4 A5 13 4A 4A 4A
.:14E8 4A AA BD C8 11 85 1D A9
.:14F0 00 85 3C A5 3C F0 FC A9
.:14F8 20 85 04 8D 12 D4 A5 13
.:1500 18 69 10 85 13 8D 13 D4
.:1508 29 F0 D0 D0 A9 A1 85 22
.:1510 A9 10 85 23 20 09 19 A9
.:1518 01 85 5D A9 50 85 13 8D
.:1520 13 D4 A9 03 85 16 8D 14
.:1528 D4 A9 C3 85 09 A9 10 85
.:1530 0A A5 09 8D 0E D4 A5 0A
.:1538 8D 0F D4 A9 21 85 04 8D
.:1540 12 D4 A5 13 29 0F AA BD
.:1548 D8 11 85 1D A9 00 85 3C
.:1550 A5 3C F0 FC A9 20 85 04
.:1558 8D 12 D4 A5 13 18 69 01
.:1560 85 13 8D 13 D4 29 0F D0
.:1568 D2 A9 B5 85 22 A9 10 85
.:1570 23 20 09 19 A9 20 8D 13
.:1578 D4 85 13 A9 00 85 16 8D
.:1580 14 D4 A9 D1 85 09 A9 12
.:1588 85 0A A5 09 8D 0E D4 A5
.:1590 0A 8D 0F D4 A9 21 85 04
.:1598 8D 12 D4 A9 01 85 1D A9
.:15A0 00 85 3C A5 3C F0 FC A9
.:15A8 20 85 04 8D 12 D4 A5 16
.:15B0 18 69 10 85 16 8D 14 D4
.:15B8 29 F0 D0 D8 A9 CC 85 22
.:15C0 A9 10 85 23 20 09 19 A9
.:15C8 F0 85 16 8D 14 D4 A9 1F
.:15D0 85 09 A9 15 85 0A A5 09
.:15D8 8D 0E D4 A5 0A 8D 0F D4
.:15E0 A9 21 85 04 8D 12 D4 A9
.:15E8 01 85 1D A9 00 85 3C A5
.:15F0 3C F0 FC A9 20 85 04 8D
.:15F8 12 D4 A5 16 29 0F AA BD
.:1600 D8 11 85 1D A9 00 85 3C
.:1608 A5 3C F0 FC A5 16 18 69
.:1610 01 85 16 8D 14 D4 29 0F
.:1618 D0 C6 A9 E2 85 22 A9 10
.:1620 85 23 20 09 19 A9 1F 85
.:1628 19 8D 18 D4 A9 06 8D 17
.:1630 D4 85 1A A9 00 85 09 A9
.:1638 00 85 0A A5 09 8D 0E D4
.:1640 A5 0A 8D 0F D4 A9 00 85
.:1648 07 A9 00 85 08 A5 07 8D
.:1650 07 D4 A5 08 8D 08 D4 A9
.:1658 00 85 17 8D 15 D4 A9 08
.:1660 85 18 8D 16 D4 A9 00 85
.:1668 13 8D 13 D4 8D 0C D4 85
.:1670 12 A9 F0 85 16 8D 14 D4
.:1678 85 15 8D 0D D4 A9 21 85
.:1680 04 8D 12 D4 85 03 8D 0B
.:1688 D4 A9 00 85 3B A5 3B F0
.:1690 FC A5 09 18 69 20 85 09
.:1698 A9 00 65 0A 85 0A A5 07
.:16A0 38 E9 20 85 07 A5 08 E9
.:16A8 00 85 08 A5 07 8D 07 D4
.:16B0 A5 08 8D 08 D4 A5 09 8D
.:16B8 0E D4 A5 0A 8D 0F D4 A5
.:16C0 09 05 0A D0 C4 A9 FA 85
.:16C8 22 A9 10 85 23 20 09 19
.:16D0 A9 00 85 09 A9 00 85 0A
.:16D8 A9 00 85 07 A9 00 85 08
.:16E0 A5 09 8D 0E D4 A5 07 8D
.:16E8 07 D4 A5 0A 8D 0F D4 A5
.:16F0 08 8D 08 D4 A9 4F 85 19
.:16F8 8D 18 D4 A9 00 85 3B A5
.:1700 3B F0 FC A5 09 18 69 20
.:1708 85 09 A9 00 65 0A 85 0A
.:1710 A5 07 38 E9 20 85 07 A5
.:1718 08 E9 00 85 08 A5 07 8D
.:1720 07 D4 A5 08 8D 08 D4 A5
.:1728 09 8D 0E D4 A5 0A 8D 0F
.:1730 D4 A5 09 05 0A D0 C4 A9
.:1738 12 85 22 A9 11 85 23 20
.:1740 09 19 A9 00 85 09 A9 00
.:1748 85 0A A9 00 85 09 A9 00
.:1750 85 0A A5 09 8D 0E D4 A5
.:1758 07 8D 07 D4 A5 0A 8D 0F
.:1760 D4 A5 08 8D 08 D4 A9 2F
.:1768 85 19 8D 18 D4 A9 00 85
.:1770 3B A5 3B F0 FC A5 09 18
.:1778 69 20 85 09 A9 00 65 0A
.:1780 85 0A A5 07 38 E9 20 85
.:1788 07 A5 08 E9 00 85 08 A5
.:1790 07 8D 07 D4 A5 08 8D 08
.:1798 D4 A5 09 8D 0E D4 A5 0A
.:17A0 8D 0F D4 A5 09 05 0A D0
.:17A8 C4 85 04 8D 12 D4 85 03
.:17B0 8D 0B D4 A9 2C 85 22 A9
.:17B8 11 85 23 20 09 19 A9 81
.:17C0 85 04 8D 12 D4 A9 80 8D
.:17C8 16 D4 85 18 A9 32 85 09
.:17D0 A9 19 85 0A A5 09 8D 0E
.:17D8 D4 A5 0A 8D 0F D4 A9 1F
.:17E0 85 19 8D 18 D4 A9 01 85
.:17E8 1D A9 00 85 3C A5 3C F0
.:17F0 FC A5 1A 18 69 10 85 1A
.:17F8 8D 17 D4 29 F0 D0 E6 A9
.:1800 3F 85 22 A9 11 85 23 20
.:1808 09 19 A9 77 85 11 8D 05
.:1810 D4 85 14 8D 06 D4 85 13
.:1818 8D 13 D4 85 16 8D 14 D4
.:1820 85 15 8D 0D D4 85 12 8D
.:1828 0C D4 A9 3E 85 09 A9 01
.:1830 85 0A A5 09 8D 0E D4 A5
.:1838 0A 8D 0F D4 A5 19 09 80
.:1840 8D 18 D4 A9 20 85 04 8D
.:1848 12 D4 A9 14 85 02 A9 31
.:1850 8D 14 03 A9 19 8D 15 03
.:1858 A9 E8 85 26 A9 11 85 27
.:1860 A9 2A 85 2C A9 12 85 2D
.:1868 A0 00 B1 26 85 05 8D 00
.:1870 D4 B1 2C 85 1B C8 B1 26
.:1878 85 06 8D 01 D4 A5 2C 18
.:1880 69 01 85 2C A9 00 65 2D
.:1888 85 2D A5 26 18 69 02 85
```

```
.:1890 26 A9 00 65 27 85 27 A9
.:1898 15 85 02 8D 04 D4 A9 00
.:18A0 85 3B A5 3B F0 FC A9 65
.:18A8 85 22 A9 11 85 23 20 09
.:18B0 19 A9 22 85 02 8D 04 D4
.:18B8 A9 E8 85 26 A9 11 85 27
.:18C0 A9 2A 85 2C A9 12 85 2D
.:18C8 A0 00 B1 26 85 05 8D 00
.:18D0 D4 B1 2C 85 1B C8 B1 26
.:18D8 85 06 8D 01 D4 A5 2C 18
.:18E0 69 01 85 2C A9 00 65 2D
.:18E8 85 2D A5 26 18 69 02 85
.:18F0 26 A9 00 65 27 85 27 A9
.:18F8 23 85 02 8D 04 D4 A9 00
.:1900 85 3B A5 3B F0 FC 4C 4B
.:1908 12 A9 90 85 24 A9 05 85
.:1910 25 A0 00 A9 20 91 24 C8
.:1918 C0 28 D0 F9 A9 90 85 24
.:1920 A9 05 85 25 A0 00 B1 22
.:1928 F0 06 91 24 C8 4C 26 19
.:1930 60 78 E6 38 E6 39 A5 39
.:1938 49 3C D0 04 85 39 E6 3A
.:1940 A5 1B F0 4E C6 1B D0 4A
.:1948 A5 02 29 FE 85 02 8D 04
.:1950 D4 A0 00 B1 26 D0 07 A9
.:1958 01 85 3B 4C 92 19 85 05
.:1960 8D 00 D4 B1 2C 85 1B C8
.:1968 B1 26 85 06 8D 01 D4 A5
.:1970 2C 18 69 01 85 2C A9 00
.:1978 65 2D 85 2D A5 26 18 69
.:1980 02 85 26 A9 00 65 27 85
.:1988 27 A5 02 09 01 85 02 8D
.:1990 04 D4 A5 1D F0 4E C6 1D
.:1998 D0 4A A9 10 85 04 8D 12
.:19A0 D4 A0 00 B1 2A D0 07 A9
.:19A8 01 85 3B 4C E4 19 8D 0E
.:19B0 D4 85 09 C8 B1 2A 8D 0F
.:19B8 D4 85 0A A5 2A 18 69 02
.:19C0 85 2A A9 00 65 2B 85 2B
.:19C8 A0 00 B1 30 85 1D A5 30
.:19D0 18 69 01 85 30 A9 00 65
.:19D8 31 85 31 A5 5D 09 01 85
.:19E0 04 8D 12 D4 A9 FB 8D 12
.:19E8 D0 AD 11 D0 29 7F 8D 11
.:19F0 D0 A9 FF 8D 19 D0 68 A8
.:19F8 68 AA 68 40 EA 78 E6 38
.:1A00 E6 39 A5 39 49 3C D0 10
.:1A08 85 39 E6 3A A5 1D F0 08
.:1A10 C6 1D D0 04 A9 01 85 3C
.:1A18 A9 01 85 3B A5 5D C9 01
.:1A20 F0 09 A9 40 8D 12 D4 A9
.:1A28 41 85 04 A9 FB 8D 12 D0
.:1A30 AD 11 D0 29 7F 8D 11 D0
.:1A38 A9 FF 8D 19 D0 68 A8 68
.:1A40 AA 68 40 EA 48 49 00 5F
.
.
```

Listing C-7: The COMMON Source Code

```
1000 ;PUT"@0:COMMON"
1010 ;
1020 ;THIS IS COMMON--THE STANDARD
1030 ;ADDITIONS FOR COMMODORE MUSIC
1040 ;  10/29/83
1050 ;NOTE DEFINITIONS
1060 ;
1070 CN0    = $010C
1080 CS0    = $011C
1090 DN0    = $012D
1100 DS0    = $013E
1110 EN0    = $0151
1120 FN0    = $0166
1130 FS0    = $017B
1140 GN0    = $0191
1150 GS0    = $01A9
1160 AN0    = $01C3
1170 AS0    = $01DD
1180 BN0    = $01FA
1190 CN1    = $0218
1200 CS1    = $0238
1210 DN1    = $025A
1220 DS1    = $027D
1230 EN1    = $02A3
1240 FN1    = $02CC
1250 FS1    = $02F6
1260 GN1    = $0323
1270 GS1    = $0353
1280 AN1    = $0386
1290 AS1    = $03BB
1300 BN1    = $03F4
1310 CN2    = $0430
1320 CS2    = $0470
1330 DN2    = $04B4
1340 DS2    = $04F6
1350 EN2    = $0547
1360 FN2    = $0598
1370 FS2    = $05ED
1380 GN2    = $0647
1390 GS2    = $06A7
1400 AN2    = $070C
1410 AS2    = $0777
1420 BN2    = $07E9
1430 CN3    = $0861
1440 CS3    = $08E1
1450 DN3    = $0968
1460 DS3    = $09F7
1470 EN3    = $0A8F
1480 FN3    = $0B30
1490 FS3    = $0BDA
1500 GN3    = $0C8F
1510 GS3    = $0D4E
```

```
1520 AN3    = $0E18
1530 AS3    = $0EEF
1540 BN3    = $0FD2
1550 CN4    = $10C3
1560 CS4    = $11C3
1570 DN4    = $12D1
1580 DS4    = $13EF
1590 EN4    = $151F
1600 FN4    = $1660
1610 FS4    = $17B5
1620 GN4    = $1932
1630 GS4    = $1A9C
1640 AN4    = $1C31
1650 AS4    = $1DDF
1660 BN4    = $1FA5
1670 CN5    = $2187
1680 CS5    = $2386
1690 DN5    = $25A2
1700 DS5    = $27DF
1710 EN5    = $2A3E
1720 FN5    = $2CC1
1730 FS5    = $2F6B
1740 GN5    = $323C
1750 GS5    = $3539
1760 AN5    = $3863
1770 AS5    = $3BBE
1780 BN5    = $3F4B
1790 CN6    = $430F
1800 CS6    = $470C
1810 DN6    = $4B45
1820 DS6    = $4FBF
1830 EN6    = $547D
1840 FN6    = $5983
1850 FS6    = $5ED6
1860 GN6    = $6479
1870 GS6    = $6A73
1880 AN6    = $70C7
1890 AS6    = $777C
1900 BN6    = $7E97
1910 CN7    = $861E
1920 CS7    = $8E18
1930 DN7    = $968B
1940 DS7    = $9F7E
1950 EN7    = $A8FA
1960 FN7    = $B306
1970 FS7    = $BDAC
1980 GN7    = $C8F3
1990 GS7    = $D4E6
2000 AN7    = $E18F
2010 AS7    = $EEF8
2020 BN7    = $FD2E
2030 ;
2040 ;NOTE TIME CHART
2050 ;
2060 WHL    = $3C  ;WHOLE NOTE
2070 HLF    = $1E  ;HALF NOTE
2080 QRT    = $0F  ;QUARTER NOTE
2090 TRP    = $14  ;TRIPLET
2100 EGT    = $07  ;EIGHTH NOTE
2110 SXT    = $03  ;SIXTEENTH NOTE
2120 NTH    = $06  ;NINTH NOTE-1/3
                    TRIPLET
2130 DHF    = $2D  ;DOTTED HALF
2140 DQR    = $16  ;DOTTED QUARTER
2150 DEG    = $0A  ;DOTTED EIGHTH
2160 DSX    = $04  ;DOTTED SIXTEENTH
2170 ;
2180 ;
2190 ;
2200        .END
```

Listing C-8: The SNDDEF Source Code

```
1000 ;PUT"@0:SNDDEF"
1010 ;
1020 ;RAM DEFINITIONS
1030 ;
1040 *      = $02
1050 S1CORG DS 1
1060 S2CORG DS 1
1070 S3CORG DS 1
1080 S1FRLO DS 1
1090 S1FRHI DS 1
1100 S2FRLO DS 1
1110 S2FRHI DS 1
1120 S3FRLO DS 1
1130 S3FRHI DS 1
1140 S1PWLO DS 1
1150 S1PWHI DS 1
1160 S2PWLO DS 1
1170 S2PWHI DS 1
1180 S3PWLO DS 1
1190 S3PWHI DS 1
1200 S1ATDC DS 1
1210 S2ATDC DS 1
1220 S3ATDC DS 1
1230 S1SURL DS 1
1240 S2SURL DS 1
1250 S3SURL DS 1
1260 FILLO  DS 1
1270 FILHI  DS 1
1280 MMOD   DS 1
1290 RFIL   DS 1
1300 SNDTM1 DS 1
1310 SNDTM2 DS 1
1320 SNDTM3 DS 1
1330 SCRPT  DS 2
1340 SCRDST DS 2
1350 SRC    DS 2
1360 DST    DS 2
1370 NOTPT1 DS 2
1380 NOTPT2 DS 2
1390 NOTPT3 DS 2
```

```
1400 NOTTM1 DS 2
1410 NOTTM2 DS 2
1420 NOTTM3 DS 2
1430 OPTION DS 1
1440 RAND1  DS 1
1450 RAND2  DS 1
1460 RAND3  DS 1
1470 RAND4  DS 1
1480 LEVEL  DS 1
1490 ;
1500 SCREEN DS 1
1510 RANSEC DS 1
1520 SECOND DS 1
1530 ENABLE DS 1
1540 TIMOUT DS 1
1550 ;
1560 BUF    DS 8
1570 BUF1   DS 8
1580 IBUF   DS 8
1590 MBUF   DS 8
1600 ;
1610 WAVTYP DS 1
1620 ;
1630 CURS   DS 1
1640 NYB    DS 1
1650 BASE   DS 2
1660 ;
1670 ;
1680        .END
```

Listing C-9: The DATA Source Code

```
1000 ;PUT"@0:DATA"
1010 ;
1020 ;PUT THE DATA HERE
1030 ;
1040 RET    RTI
1050        .BYTE '(C) COPYRIGHT 1984, STEVEN BRESS'
1060 ;
1070 MTRI   .BYTE 'TRIANGLE WAVEFORM'
1080        .BYTE $00
1090 MSAW   .BYTE 'SAWTOOTH WAVEFORM'
1100        .BYTE $00
1110 MSQU   .BYTE 'SQUARE WAVEFORM'
1120        .BYTE $00
1130 MNOI   .BYTE 'NOISE WAVEFORM'
1140        .BYTE $00
1150 MCPW   .BYTE 'CHANGING PULSEWIDTH'
1160        .BYTE $00
1170 MOCT   .BYTE 'CHANGING OCTAVES'
1180        .BYTE $00
1190 MATT   .BYTE 'CHANGING ATTACK TIME'
1200        .BYTE $00
1210 MDEC   .BYTE 'CHANGING DECAY TIME'
1220        .BYTE $00
1230 MSUS   .BYTE 'CHANGING SUSTAIN LEVEL'
1240        .BYTE $00
1250 MREL   .BYTE 'CHANGING RELEASE TIME'
1260        .BYTE $00
1270 MFLO   .BYTE 'CHANGING LO-PASS FILTER'
1280        .BYTE $00
1290 MFHI   .BYTE 'CHANGING HI-PASS FILTER'
1300        .BYTE $00
1310 MFBA   .BYTE 'CHANGING BAND-PASS FILTER'
1320        .BYTE $00
1330 MRES   .BYTE 'CHANGING RESONANCE'
1340        .BYTE $00
1350 MRMO   .BYTE 'RING MODULATION ON'
1360        .BYTE $00
1370 MRMC   .BYTE 'CHANGING MODULATOR'
1380        .BYTE $00
1390 MSYN   .BYTE 'SYNCHRONIZATION ON'
```

```
1400          .BYTE $00
1410 MSYS     .BYTE 'CHANGING SYNC FREQUENCIES'
1420          .BYTE $00
1430 ;
1440 OCTAVE .WORD CN0,CN1,CN2,CN3,CN4,CN5,CN6,CN7,$0000
1450 ;
1460 OCTTM    .BYTE WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,$00
1470 ;
1480 TUNES    .WORD AN2,BN2,CN3,GN3,FN3,EN3,DN3,CN3,$0000
1490 TUNTM    .BYTE QRT,QRT,QRT,QRT,QRT,QRT,QRT,QRT,$00
1500 ;
1510 ATLK     .BYTE $01,$01,$01,$01,$01,$01,$01,$01,$01,$02,$02,$02,$02,$04,
              $06,$09
1520 DCLK     .BYTE $01,$01,$01,$01,$01,$01,$01,$01,$01,$02,$02,$03,$04,$0A,
              $10,$1A
1530 ;
1540 MODLK    .WORD CN0,EN0,GN0,AN0,CN1,EN1,GN1,AN1,CN2,EN2,GN2,AN2
1550          .WORD CN3,EN3,GN3,AN3,CN4,EN4,GN4,AN4,CN5,EN5,GN5,AN5
1560          .WORD CN6,EN6,GN6,AN6,CN7,EN7,GN7,AN7,$0000
1570 MODTM    .BYTE WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL
1580          .BYTE WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL
1590          .BYTE WHL,WHL,WHL,WHL,WHL,WHL,WHL,WHL,$00
1600 ;
1610          .END
```

Listing C-10: The EDIT SND Program, Source Code for the SOUND EDIT Program

```
1000 ;PUT"@0:EDIT SND"
1010 ;LOAD"ASM",8
1020 ;
1030 ;THIS IS A SOUND EFFECTS DEMO
1040 ;
1050 ;(C) COPYRIGHT 1984, STEVEN BRESS
1060 ;
1070 ;CREATED 7/9/84
1080 ;LATEST ADDITIONS 7/16/84
1090 ;
1100          .OPT LIST,NOSYM,NOGEN
1110 ;
1120 ;
1130          .LIB MACLIB
1140          .LIB SYSDEF
1150          .LIB COMMON
1160          .LIB SNDDEF
1170 ;
1180 *        = $1000
1190          JMP TOP
1200 ;
1210 RET      RTI
1220 CURSH    .WORD $000A,$0017,$0024,$005A,$0067,$0074,$00AA,$00B7,$00C4
1230          .WORD $00FA,$0107,$0114,$014A,$0157,$0164,$019A,$01A7,$01B4,
              $01EA
1240          .WORD $01F7,$0204,$023A,$0247,$0254,$028A,$0297,$02A4,$0000,
              $0000
1250 STMR     .WORD $03AF
1260 ;
1270 REGI     .BYTE S1ATDC,S2ATDC,S3ATDC,S1SURL,S2SURL,S3SURL,S1FRLO,S2FRLO
1280          .BYTE S3FRLO,S1FRHI,S2FRHI,S3FRHI,S1PWLO,S2PWLO,S3PWLO,S1PWHI
1290          .BYTE S2PWHI,S3PWHI,S1CORG,S2CORG,S3CORG,FILLO,FILHI,MMOD
```

```
1300          .BYTE RFIL,SNDTM1,SNDTM2
1310 ;
1320 TOP      ANOP
1330          KILL
1340          MVCOL WHITE
1350          LDMEM #BLUE,BORCL
1360          STA BCOL0
1370          LDMEM #$01,SPREN
1380          STA BPRIOR
1390          STA SPRCL0
1400          ADRES $0400,BASE
1410          LDMEM #$00,$DC03
1420          LDMEM #$FF,$DC02
1430          LDMEM #$7F,$DC00
1440 ;
1450          MVMEM $2000,$0400,$04
1460 ;
1470          ADRES INT0,CINV
1480          RAST #$FB
1490          CLI
1500 ;
1510 START    ANOP
1520          LDA CURS
1530          ASL A
1540          TAX
1550          LDA CURSH,X
1560          STA BUF
1570          LDA CURSH+1,X
1580          STA BUF+1
1590          LDA BUF
1600          ORA BUF+1
1610          BNE ST1
1620          LDMEM #$00,CURS
1630          JMP START
1640 ;
1650 ST1      LDA BUF+1
1660          LDY NYB
1670          CLC
1680          ADC #$D8         ;COLOR OFFSET
1690          STA BUF+1
1700 ;
1710          LDA SCREEN
1720          AND #$10
1730          BEQ CRWT
1740          LDA #CYAN
1750          JMP PCLR1
1760 ;
1770 CRWT     LDA #WHITE
1780 PCLR1    STA (BUF),Y
1790 ;
1800 ;
1810          LDA SCREEN
1820          AND #$10
1830          JNE CHDN2
1840          JSR $FFE4
1850 ;
1860          STA BUF
1870          JEQ CHDN2
1880          CMP #$1D
```

```
1890          JEQ CRRT          ;CURSORS
1900          CMP #$11
1910          JEQ CRDN
1920          CMP #$91
1930          JEQ CRUP
1940          CMP #$9D
1950          JEQ CRLF
1960          CMP #$85          ;FUNCTION 1
1970          JEQ TRIG
1980          CMP #$47          ;G
1990          JCS CHDN2
2000          CMP #$41          ;A
2010          BCS CHOK3
2020          CMP #$3A          ;:==9+1
2030          JCS CHDN2
2040          CMP #$30          ;0
2050          BCS CHOK1
2060          JMP CHDN2
2070 CHOK3    ANOP
2080          AND #$0F
2090 ;
2100 ;
2110 ;
2120 CHOK1    ANOP
2130          STA BUF1
2140 ;
2150 CHOK2    ANOP
2160          LDA CURS
2170          ASL A
2180          TAX
2190          LDA CURSH,X
2200          STA BUF
2210          LDA CURSH+1,X
2220          STA BUF+1
2230 ;
2240          LDA BUF
2250          ORA BUF+1
2260          BNE PUCH
2270          LDMEM #$00,CURS
2280          JMP CHOK2
2290 ;
2300 PUCH     LDY NYB
2310          DBADC BASE,BUF,BUF
2320          LDA BUF1
2330          STA (BUF),Y
2340          LDA BUF+1
2350          CLC
2360          ADC #$D4          ;PUTS DATA AT $D800
2370          STA BUF+1
2380          LDA #WHITE
2390          STA (BUF),Y
2400          CPY #$00
2410          BNE NXRO
2420          INY
2430          STY NYB
2440          JMP CHDN2
2450 ;
2460 NXRO     LDMEM #$00,NYB
2470          INC CURS
```

```
        JMP CHDN2
;
CRUP    ANOP
        LDMEM #$00,NYB
        LDA CURS
        SEC
        SBC #$03
        STA CURS
        BPL CHDN2
;
        LDMEM #$1C,CURS
        JMP CHDN2
;
CRDN    ANOP
        LDMEM #$00,NYB
        LDA CURS
        CLC
        ADC #$03
        STA CURS
        CMP #$1D
        BCC CHDN2
;
        LDMEM #$00,CURS
        JMP CHDN2
;
CRRT    ANOP
        INC CURS
        LDMEM #$00,NYB
        LDA CURS
        CMP #$1D
        BCC CHDN2
;
        LDMEM #$00,CURS
        JMP CHDN2
;
CRLF    ANOP
        LDMEM #$00,NYB
        DEC CURS
        BPL CHDN2
;
        LDMEM #$1C,CURS
;
CHDN2   ANOP
;
;
;CHANGE CURSOR COLOR
;
;
;
;
;
        LDMEM #$00,ENABLE
TWID    LDA ENABLE
        BEQ TWID
        JMP START
;
;
INT0    SEI
        INC SCREEN
```

```
3070          INC RANSEC
3080          LDA RANSEC
3090          EOR #$3C
3100          BNE SYNC1
3110 ;
3120          STA RANSEC
3130          INC SECOND
3140 SYNC1    ANOP
3150 ;
3160          LDA SNDTM1
3170          ORA SNDTM1+1
3180          BEQ SNDDN
3190          DBDEC SNDTM1,1
3200          LDA SNDTM1
3210          ORA SNDTM1+1
3220          BNE SNDDN
3230 ;
3240          LDA S1CORG
3250          AND #$FE
3260          STA S1CORG
3270          STA V1CORG
3280 ;
3290          LDA S2CORG
3300          AND #$FE
3310          STA S2CORG
3320          STA V2CORG
3330 ;
3340          LDA S3CORG
3350          AND #$FE
3360          STA S3CORG
3370          STA V3CORG
3380 ;
3390 SNDDN    ANOP
3400 ;
3410          JSR $FF9F          ;SCAN KEY
3420 ;
3430          LDMEM #$01,ENABLE
3440          RAST #$FB
3450          PULL
3460 ;
3470 ;
3480 ;
3490 TRIG     ANOP
3500          LDX #$00
3510 TRGLP    TXA
3520          ASL A
3530          TAY
3540          LDA CURSH,Y
3550          STA BUF
3560          LDA CURSH+1,Y
3570          STA BUF+1
3580          DBADC BASE,BUF,BUF
3590          LDY #$00
3600          LDA (BUF),Y
3610          JSR ASCBIN
3620          NIBLL
3630          STA BUF1
3640          LDY #$01
3650          LDA (BUF),Y
```

```
        JSR ASCBIN
        ORA BUF1
        STA BUF1
;
        LDY REGI,X
        STA $00,Y
        INX
        CPX #$1B
        BNE TRGLP
;
        LDMEM S1ATDC,V1ATDC
        LDMEM S2ATDC,V2ATDC
        LDMEM S3ATDC,V3ATDC
        LDMEM S1SURL,V1SURL
        LDMEM S2SURL,V2SURL
        LDMEM S3SURL,V3SURL
        LDMEM S1FRLO,V1FRLO
        LDMEM S2FRLO,V2FRLO
        LDMEM S3FRLO,V3FRLO
        LDMEM S1FRHI,V1FRHI
        LDMEM S2FRHI,V2FRHI
        LDMEM S3FRHI,V3FRHI
        LDMEM S1PWLO,V1PWLO
        LDMEM S2PWLO,V2PWLO
        LDMEM S3PWLO,V3PWLO
        LDMEM S1PWHI,V1PWHI
        LDMEM S2PWHI,V2PWHI
        LDMEM S3PWHI,V3PWHI
        LDMEM FILLO,FLCNLO
        LDMEM FILHI,FLCNHI
        LDMEM MMOD,MODVOL
        LDMEM RFIL,RESFLT
        LDMEM S1CORG,V1CORG
        LDMEM S2CORG,V2CORG
        LDMEM S3CORG,V3CORG
;
        JMP CHDN2
;
;
ASCBIN  ANOP
;
        STA BUF1+2
        CMP #$07          ;IS IT A LETTER
        BCS NUM
        CLC
        ADC #$09          ;DROP IT TO NUMBER
        RTS
;
NUM     AND #$0F
        RTS
;
;
BINASC  ANOP
;
;
;
        RTS
;
        NOP
```

```
4250          NOP
4260          .END
```

Listing C-11: The SOUND EDIT Program

```
B*
   PC  SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:1000 4C 5B 10 40 0A 00 17 00
.:1008 24 00 5A 00 67 00 74 00
.:1010 AA 00 B7 00 C4 00 FA 00
.:1018 07 01 14 01 4A 01 57 01
.:1020 64 01 9A 01 A7 01 B4 01
.:1028 EA 01 F7 01 04 02 3A 02
.:1030 47 02 54 02 8A 02 97 02
.:1038 A4 02 00 00 00 00 AF 03
.:1040 11 12 13 14 15 16 05 07
.:1048 09 06 08 0A 0B 0D 0F 0C
.:1050 0E 10 02 03 04 17 18 19
.:1058 1A 1B 1C 78 A9 06 85 01
.:1060 A9 00 8D 0D DC 8D 0D DD
.:1068 AD 0D DC AD 0D DD A9 00
.:1070 8D 02 DC 8D 03 DC 8D 0E
.:1078 DC 8D 0F DC 8D 0E DD 8D
.:1080 0F DD A9 01 8D 1A D0 A9
.:1088 FF 8D 19 D0 A2 02 A9 00
.:1090 95 00 E8 D0 FB A2 FF 9A
.:1098 A9 03 8D 18 03 A9 10 8D
.:10A0 19 03 A9 00 85 20 A9 D8
.:10A8 85 21 A0 00 A2 00 A9 01
.:10B0 91 20 C8 D0 FB E6 21 A5
.:10B8 21 C9 DC D0 F1 A9 06 8D
.:10C0 20 D0 8D 21 D0 A9 01 8D
.:10C8 15 D0 8D 1B D0 8D 27 D0
.:10D0 A9 00 85 60 A9 04 85 61
.:10D8 A9 00 8D 03 DC A9 FF 8D
.:10E0 02 DC A9 7F 8D 00 DC A9
.:10E8 00 85 1E A9 20 85 1F A9
.:10F0 00 85 20 A9 04 85 21 A0
.:10F8 00 A2 00 B1 1E 91 20 C8
.:1100 D0 F9 E6 1F E6 21 E8 E0
.:1108 04 D0 F0 A9 49 8D 14 03
.:1110 A9 12 8D 15 03 A9 FB 8D
.:1118 12 D0 AD 11 D0 29 7F 8D
.:1120 11 D0 58 A5 5E 0A AA BD
.:1128 04 10 85 3D BD 05 10 85
.:1130 3E A5 3D 05 3E D0 07 A9
.:1138 00 85 5E 4C 23 11 A5 3E
.:1140 A4 5F 18 69 D8 85 3E A5
.:1148 38 29 10 F0 05 A9 03 4C
.:1150 54 11 A9 01 91 3D A5 38
.:1158 29 10 F0 03 4C 3E 12 20
.:1160 E4 FF 85 3D D0 03 4C 3E
.:1168 12 C9 1D D0 03 4C 1F 12
.:1170 C9 11 D0 03 4C 09 12 C9
.:1178 91 D0 03 4C F5 11 C9 9D
.:1180 D0 03 4C 32 12 C9 85 D0
.:1188 03 4C AB 12 C9 47 90 03
.:1190 4C 3E 12 C9 41 B0 0E C9
.:1198 3A 90 03 4C 3E 12 C9 30
.:11A0 B0 05 4C 3E 12 29 0F 85
.:11A8 45 A5 5E 0A AA BD 04 10
.:11B0 85 3D BD 05 10 85 3E A5
.:11B8 3D 05 3E D0 07 A9 00 85
.:11C0 5E 4C A9 11 A4 5F A5 60
.:11C8 18 65 3D 85 3D A5 61 65
.:11D0 3E 85 3E A5 45 91 3D A5
.:11D8 3E 18 69 D4 85 3E A9 01
.:11E0 91 3D C0 00 D0 06 C8 84
.:11E8 5F 4C 3E 12 A9 00 85 5F
.:11F0 E6 5E 4C 3E 12 A9 00 85
.:11F8 5F A5 5E 38 E9 03 85 5E
.:1200 10 3C A9 1C 85 5E 4C 3E
.:1208 12 A9 00 85 5F A5 5E 18
.:1210 69 03 85 5E C9 1D 90 26
.:1218 A9 00 85 5E 4C 3E 12 E6
.:1220 5E A9 00 85 5F A5 5E C9
.:1228 1D 90 13 A9 00 85 5E 4C
.:1230 3E 12 A9 00 85 5F C6 5E
.:1238 10 04 A9 1C 85 5E A9 00
.:1240 85 3B A5 3B F0 FC 4C 23
.:1248 11 78 E6 38 E6 39 A5 39
.:1250 49 3C D0 04 85 39 E6 3A
.:1258 A5 1B 05 1C F0 2E A5 1B
.:1260 38 E9 01 85 1B A5 1C E9
.:1268 00 85 1C A5 1B 05 1C D0
.:1270 1B A5 02 29 FE 85 02 8D
.:1278 04 D4 A5 03 29 FE 85 03
.:1280 8D 0B D4 A5 04 29 FE 85
.:1288 04 8D 12 D4 20 9F FF A9
.:1290 01 85 3B A9 FB 8D 12 D0
.:1298 AD 11 D0 29 7F 8D 11 D0
.:12A0 A9 FF 8D 19 D0 68 A8 68
.:12A8 AA 68 40 A2 00 8A 0A A8
.:12B0 B9 04 10 85 3D B9 05 10
.:12B8 85 3E A5 60 18 65 3D 85
.:12C0 3D A5 61 65 3E 85 3E A0
.:12C8 00 B1 3D 20 6A 13 0A 0A
.:12D0 0A 0A 85 45 A0 01 B1 3D
.:12D8 20 6A 13 05 45 85 45 BC
.:12E0 40 10 99 00 00 E8 E0 1B
.:12E8 D0 C3 A5 11 8D 05 D4 A5
.:12F0 12 8D 0C D4 A5 13 8D 13
.:12F8 D4 A5 14 8D 06 D4 A5 15
.:1300 8D 0D D4 A5 16 8D 14 D4
.:1308 A5 05 8D 00 D4 A5 07 8D
```

```
.:1310 07 D4 A5 09 8D 0E D4 A5
.:1318 06 8D 01 D4 A5 08 8D 08
.:1320 D4 A5 0A 8D 0F D4 A5 0B
.:1328 8D 02 D4 A5 0D 8D 09 D4
.:1330 A5 0F 8D 10 D4 A5 0C 8D
.:1338 03 D4 A5 0E 8D 0A D4 A5
.:1340 10 8D 11 D4 A5 17 8D 15
.:1348 D4 A5 18 8D 16 D4 A5 19
.:1350 8D 18 D4 A5 1A 8D 17 D4
.:1358 A5 02 8D 04 D4 A5 03 8D
.:1360 0B D4 A5 04 8D 12 D4 4C
.:1368 3E 12 85 47 C9 07 B0 04
.:1370 18 69 09 60 29 0F 60 60
.:1378 EA EA EC 01 05 F0 41 59
.
.

.:2000 20 20 16 31 01 14 04 03
.:2008 3D 24 30 30 20 20 20 16
.:2010 02 01 14 04 03 3D 24 30
.:2018 30 20 20 20 16 33 01 14
.:2020 04 03 3D 24 30 30 20 20
.:2028 20 20 20 20 20 20 20 20
.:2030 20 20 20 20 20 20 20 20
.:2038 20 20 20 20 20 20 20 20
.:2040 20 20 20 20 20 20 20 20
.:2048 20 20 20 20 20 20 20 20
.:2050 20 20 16 31 13 15 12 0C
.:2058 3D 24 30 30 20 20 20 16
.:2060 32 13 15 12 0C 3D 24 30
.:2068 30 20 20 20 16 33 13 15
.:2070 12 0C 3D 24 30 30 20 20
.:2078 20 20 20 20 20 20 20 20
.:2080 20 20 20 20 20 20 20 20
.:2088 20 20 20 20 20 20 20 20
.:2090 20 20 20 20 20 20 20 20
.:2098 20 20 20 20 20 20 20 20
.:20A0 20 20 16 31 06 12 0C 0F
.:20A8 3D 24 30 30 20 20 20 16
.:20B0 32 06 12 0C 0F 3D 24 30
.:20B8 30 20 20 20 16 33 06 12
.:20C0 0C 0F 3D 24 30 30 20 20
.:20C8 20 20 20 20 20 20 20 20
.:20D0 20 20 20 20 20 20 20 20
.:20D8 20 20 20 20 20 20 20 20
.:20E0 20 20 20 20 20 20 20 20
.:20E8 20 20 20 20 20 20 20 20
.:20F0 20 20 16 31 06 12 08 09
.:20F8 3D 24 30 30 20 20 20 16
.:2100 32 06 12 08 09 3D 24 30
.:2108 30 20 20 20 16 33 06 12
.:2110 08 09 3D 24 30 30 20 20
.:2118 20 20 20 20 20 20 20 20
.:2120 20 20 20 20 20 20 20 20
.:2128 20 20 20 20 20 20 20 20
.:2130 20 20 20 20 20 20 20 20
.:2138 20 20 20 20 20 20 20 20
.:2140 20 20 16 31 10 17 0C 0F
.:2148 3D 24 30 30 20 20 20 16
.:2150 32 10 17 0C 0F 3D 24 30
.:2158 30 20 20 20 16 33 10 17
.:2160 0C 0F 3D 24 30 30 20 20
.:2168 20 20 20 20 20 20 20 20
.:2170 20 20 20 20 20 20 20 20
.:2178 20 20 20 20 20 20 20 20
.:2180 20 20 20 20 20 20 20 20
.:2188 20 20 20 20 20 20 20 20
.:2190 20 20 16 31 10 17 08 09
.:2198 3D 24 30 30 20 20 20 16
.:21A0 32 10 17 08 09 3D 24 30
.:21A8 30 20 20 20 16 33 10 17
.:21B0 08 09 3D 24 30 30 20 20
.:21B8 20 20 20 20 20 20 20 20
.:21C0 20 20 20 20 20 20 20 20
.:21C8 20 20 20 20 20 20 20 20
.:21D0 20 20 20 20 20 20 20 20
.:21D8 20 20 20 20 20 20 20 20
.:21E0 20 20 16 31 03 0F 12 07
.:21E8 3D 24 30 30 20 20 20 16
.:21F0 32 03 0F 12 07 3D 24 30
.:21F8 30 20 20 20 16 33 03 0F
.:2200 12 07 3D 24 30 30 20 20
.:2208 20 20 20 20 20 20 20 20
.:2210 20 20 20 20 20 20 20 20
.:2218 20 20 20 20 20 20 20 20
.:2220 20 20 20 20 20 20 20 20
.:2228 20 20 20 20 20 20 20 20
.:2230 20 20 06 0C 03 0E 0C 0F
.:2238 3D 24 30 30 20 20 20 06
.:2240 0C 03 0E 08 09 3D 24 30
.:2248 30 20 20 20 0D 0F 04 16
.:2250 0F 0C 3D 24 30 30 20 20
.:2258 20 20 20 20 20 20 20 20
.:2260 20 20 20 20 20 20 20 20
.:2268 20 20 20 20 20 20 20 20
.:2270 20 20 20 20 20 20 20 20
.:2278 20 20 20 20 20 20 20 20
.:2280 20 20 12 05 13 06 0C 14
.:2288 3D 24 30 30 20 20 20 13
.:2290 0E 04 14 0D 31 3D 24 30
.:2298 30 20 13 0E 04 14 0D 31
.:22A0 2B 31 3D 24 30 30 20 20
.:22A8 20 20 20 20 20 20 20 20
.:22B0 20 20 20 20 20 20 20 20
.:22B8 20 20 20 20 20 20 20 20
.:22C0 20 20 20 20 20 20 20 20
.:22C8 20 20 20 20 20 20 20 20
.:22D0 20 20 20 20 20 20 20 20
.:22D8 20 20 20 20 20 20 20 20
.:22E0 20 20 20 20 20 20 20 20
.:22E8 20 20 20 20 20 20 20 20
.:22F0 20 20 20 20 20 20 20 20
.:22F8 20 20 20 20 20 20 20 20
.:2300 20 20 20 20 20 20 20 20
.:2308 20 20 20 20 20 20 20 20
.:2310 20 20 20 20 20 20 20 20
.:2318 20 20 20 20 20 20 20 20
```

```
.:2320 20 20 20 20 20 20 20 20
.:2328 20 20 20 20 20 20 20 20
.:2330 20 20 20 20 20 20 20 20
.:2338 20 20 20 20 20 20 20 20
.:2340 20 20 20 20 20 20 20 20
.:2348 20 20 20 20 20 20 20 20
.:2350 20 20 20 20 20 20 20 20
.:2358 20 20 20 20 20 20 20 20
.:2360 20 20 20 20 20 20 20 20
.:2368 20 20 20 20 20 20 20 20
.:2370 20 20 20 20 20 20 20 20
.:2378 20 20 20 20 20 20 20 20
.:2380 20 20 20 20 20 20 20 20
.:2388 20 20 20 20 20 20 20 20
.:2390 20 20 20 20 20 20 20 20
.:2398 20 20 20 20 20 20 20 20
.:23A0 20 20 20 20 20 20 20 20
.:23A8 20 20 20 20 20 20 20 20
.:23B0 20 20 20 20 20 20 20 20
.:23B8 20 20 20 20 20 20 20 20
.:23C0 20 20 20 20 20 20 20 20
.:23C8 20 20 20 20 20 20 20 20
.:23D0 20 20 20 20 20 20 20 20
.:23D8 20 20 20 20 20 20 20 20
.:23E0 20 20 20 20 20 20 20 20
.:23E8 04 F6 00 F1 00 3D 04 34
.:23F0 24 24 00 35 91 7F 94 05
.:23F8 15 95 32 05 05 80 95 FF
.:2400 7F FF 00 00 FF FF 00 00
.
.
```

Listing C-12: The KO-COM Program

```
5 GOTO 110
25 A$="R0:"+CN$+A$
30 OPEN 15,8,15,A$
40 CLOSE15
50 END
110 INPUT"WHAT IS THE KOALA FILE NAME";KN$
120 INPUT"WHAT IS THE COMMODORE FILE NAME";CN$
130 A$=CHR$(129)+LEFT$(KN$+"                ",14)
140 CN$=CN$+"="
150 GOTO 25
```

Listing C-13: The COM-KO Program

```
5 GOTO 110
25 A$="R0:"+A$+CN$
30 OPEN 15,8,15,A$
40 CLOSE15
50 END
110 INPUT"WHAT IS THE KOALA FILE NAME";KN$
120 INPUT"WHAT IS THE COMMODORE FILE NAME";CN$
130 A$=CHR$(129)+LEFT$(KN$+"               ",14)
140 CN$="="+CN$
150 GOTO 25
```

Listing C-14: The DISPLAY PIC Program

```
10 IF A=0 THEN A=1:LOAD"MVIT",8,1
20 IF A=1THEN A=2:INPUT"WHAT IS THE FILE NAME";QR$:LOAD QR$,8,1
30 IF A=2THEN A=3:FOR X=1 TO 500:NEXT:SYS8192
40 WAIT 653,1:WAIT653,1,1
60 STOP
```

Listing C-15: The MVIT Subroutine

```
B*
   PC   SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:2000 4C 13 20 2F 18 69 3D 49
.:2008 3A 32 03 18 69 07 4C D0
.:2010 27 32 10 08 48 8A 48 98
.:2018 48 A9 40 85 FC A9 7F 85
.:2020 FD A9 00 85 FE A9 5C 85
.:2028 FF A0 00 A2 00 B1 FC 91
.:2030 FE C8 D0 F9 E6 FD E6 FF
.:2038 E8 E0 04 D0 F0 A9 28 85
.:2040 FC A9 83 85 FD A9 00 85
.:2048 FE A9 D8 85 FF A0 00 A2
.:2050 00 B1 FC 91 FE C8 D0 F9
.:2058 E6 FD E6 FF E8 E0 04 D0
.:2060 F0 AD 10 87 8D 21 D0 A9
.:2068 03 8D 02 DD 38 E9 01 8D
.:2070 00 DD A9 60 4A 4A 8D 0B
.:2078 20 AD 18 D0 29 F0 0D 0B
.:2080 20 8D 18 D0 A9 5C 0A 0A
.:2088 8D 0B 20 AD 18 D0 29 0F
.:2090 0D 0B 20 8D 18 D0 AD 16
.:2098 D0 09 10 8D 16 D0 AD 11
.:20A0 D0 09 20 8D 11 D0 68 A8
.:20A8 68 AA 68 28 60 63 42 50
.
```

Listing C-16: The PIC A CASTLE Koala Pad Picture File

```
B*
   PC   SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:6000 FF FF FF FF FF FF FF FF
.:6008 7F FF FF FF FF FF FF FF
.:6010 FF FF FF FF FF FF FF FF
.:6018 FF FF FF FF FF FF FF FF
.:6020 FF FF FF FF FF FF FF FF
.:6028 FF FF FF FF FF FF FF FF
.:6030 FF FF FF FF FF FF FF FF
.:6038 FF FF FF FF FF FF FF FF
.:6040 FF FF FF FF FF FF FF FF
.:6048 FF FF FF FF FF FF FF FF
.:6050 FF FF FF FF FF FF FF FF
.:6058 FF FF FF FF FF FF FF FF
.:6060 FF FF FF FF FF FF FF FF
.:6068 FF FF FF FF FF FF FF FF
.:6070 FF FF FF FF FF FF FF FF
.:6078 FF FF FF FF FF FF FF FF
.:6080 FF FF FF FF FF FF FF FF
.:6088 FF FF FF FF FF FF FF FF
.:6090 FF FF FF FF FF FF FF FF
.:6098 FF FF FF FF FF FF FF FF
.:60A0 FF FF FF FF FF FF FF FF
.:60A8 FF FF FF FF FF FF FF FF
.:60B0 FF FF FF FF FF FF FF FF
.:60B8 FF FF FF FF FF FF FF FF
.:60C0 FF FF FF FF FF FF FF FF
.:60C8 FF FF FF FF FF FF FF 55
.:60D0 FF FF FF FF FF FF FF FF
.:60D8 FF FF FF FF FF FF FF 55
.:60E0 FF FF FF FF FF FF FF FF
.:60E8 FF FF FF FF FF FF FF 55
.:60F0 FF FF FF FF FF FF FF FF
.:60F8 FF FF FF FF FF FF FF AA
.:6100 FF FF FF FF FF FF FF FF
.:6108 FF FF FF FF FF FF FF FF
.:6110 FF FF FF FF FF FF FF FF
.:6118 FF FF FF FF FF FF FF FF
.:6120 FF FF FF FF FF FF FF FF
.:6128 FF FF FF FF FF FF FF FF
.:6130 FF FF FF FF FF FF FF FF
.:6138 FF FF FF FF FF FF FF FF
.:6140 FF FF FF FF FF FF FF FF
.:6148 FF FF FF FF FF FF FF 55
.:6150 FF FF FF FF FF FF FF AA
.:6158 FF FF FF FF FF FF FF 55
.:6160 FF FF FF FF FF FF FF FF
.:6168 FF FF FF FF FF FF FF FF
.:6170 FF FF FF FF FF FF FF AA
.:6178 FF FF FF FF FF FF FF 55
.:6180 FF FF FF FF FF FF FF AA
.:6188 55 55 55 55 55 55 55 55
.:6190 FF FF FF FF FF FF FF 55
.:6198 FF FF FF FF FF FF FF 55
.:61A0 FF FF FF FF FF FF FF FF
.:61A8 FF FF FF FF FF FF FF 55
.:61B0 FF FF FF FF FF FF FF FF
.:61B8 FF FF FF FF FF FF FF 55
.:61C0 FF FF FF FF FF FF FF FF
.:61C8 FF FF FF FF FF FF FF AA
.:61D0 FF FF FF FF FF FF FF FF
.:61D8 FF FF FF FF FF FF FF 55
.:61E0 FF FF FF FF FF FF FF FF
.:61E8 FF FF FF FF FF FF FF 55
.:61F0 FF FF FF FF FF FF FF AA
.:61F8 FF FF FF FF FF FF FF AA
```

```
.:6200 FE FE FE FE FE FE FE AA
.:6208 55 55 55 55 55 55 55 55
.:6210 FF FF FF FF FF FF D7 D7
.:6218 55 55 55 55 55 55 69 69
.:6220 FF FF FF FF FF FF FA FA
.:6228 FF FF FF FF FF FF FE FE
.:6230 FF FF FF FF FF FF BF BF
.:6238 FF FF FF FF FF FF AF AF
.:6240 FF FF FF FF FF FF FF FF
.:6248 FF FF FF FF FF FF FF FF
.:6250 55 55 55 55 55 55 55 55
.:6258 55 55 55 55 55 55 55 55
.:6260 55 55 55 55 55 55 55 55
.:6268 FF FF FF FF FF FF FF FF
.:6270 FF FF FF FF FF FF FF FF
.:6278 FF FF FF FF FF FF FF FF
.:6280 55 55 55 55 55 55 55 55
.:6288 55 55 55 55 55 55 55 55
.:6290 FF FF FF FF FF FF FF FF
.:6298 55 55 55 55 55 55 55 55
.:62A0 FF FF FF FF FF FF FF 3F
.:62A8 55 55 55 55 55 55 55 55
.:62B0 FF FF FF FF FF FF FF FC
.:62B8 55 55 55 53 50 40 00 28
.:62C0 55 55 55 55 95 15 F5 FD
.:62C8 55 55 55 55 55 55 55 55
.:62D0 55 55 55 55 55 55 55 55
.:62D8 55 55 55 55 55 55 55 55
.:62E0 AA AA AA AA AA AA AA AA
.:62E8 AA AA AA AA AA AA AA AA
.:62F0 FF FF FF FF FF FF FF FF
.:62F8 AA AA AA AA AA AA AA AA
.:6300 FF FF FF FF FF FF FF FF
.:6308 FF FF FF FF FF FF FF FF
.:6310 FF FF FF FF FF FF FF FF
.:6318 55 55 55 55 55 55 D5 35
.:6320 AA AA AA AA AA AA AA AA
.:6328 55 55 55 55 55 55 55 55
.:6330 55 55 55 55 55 55 55 55
.:6338 55 55 55 55 55 55 55 55
.:6340 55 55 55 55 55 51 50 7C
.:6348 AA AA AA AA AA AA AA 0A
.:6350 D7 D7 D7 D5 D5 D5 FD FD
.:6358 EB EB EB AA AA AA AA 8A
.:6360 FA FA FA AA AA AA AA AA
.:6368 57 57 57 FF FF FF FF FF
.:6370 BF BF BF AA AA AA 66 99
.:6378 AF AF AF AF AF AF BF FF
.:6380 FF FF FF FF FF FF FF FF
.:6388 FF FF FF FF FF FF FF 3F
.:6390 55 55 55 55 55 55 55 55
.:6398 55 55 55 55 55 55 55 54
.:63A0 55 55 55 55 55 45 01 00
.:63A8 FF FF FF FF FF FF FF FF
.:63B0 FF FF FF FF FF FF FF FF
.:63B8 FF FF FF FF FF FF FF FF
.:63C0 FF FF FF FF FF FF BF BF
.:63C8 AA AA AA AA AA AA AA AA
.:63D0 AA AA AA AA A9 A5 95 55
.:63D8 A8 A0 81 55 55 55 55 55
.:63E0 0F 03 A0 AA AA AA AA AA
.:63E8 55 55 15 A5 AB AE BA EA
.:63F0 FC FA EA AA AA AA A8 A0
.:63F8 A8 A8 A0 A2 82 0A 2A AA
.:6400 57 55 55 55 55 55 55 55
.:6408 AA AA 6A 5A 5A 56 55 55
.:6410 FF FF FF FF FF FF FF FF
.:6418 FF FF FF FC F0 F2 EA AA
.:6420 FF CF 03 00 50 54 55 55
.:6428 55 55 55 55 15 15 05 A5
.:6430 FF FF FF FF FF FF FF FF
.:6438 55 55 55 55 55 55 54 50
.:6440 FF FF FF FF FF FF FF 3F
.:6448 FF FF FF FF FF FE FE FA
.:6450 FC F0 C0 02 AA AA AA AA
.:6458 0F 03 AB AA AA AA AA AA
.:6460 FF FF FF FF FF 7F 5D 59
.:6468 55 55 54 50 40 0F FF FF
.:6470 55 55 55 15 05 F5 FE FB
.:6478 55 57 5A 6A 6A AA AA AA
.:6480 55 55 D5 75 5F 55 55 55
.:6488 03 A0 AA AA AA 56 A9 AA
.:6490 FF FF BF AF AB AA 6A 9A
.:6498 3B CE 3B CE 3B CE 3B CE
.:64A0 66 99 66 AA 6A AA 6A AA
.:64A8 BB EF BB EF FB ED FB ED
.:64B0 DD 77 DD 77 DD 77 DD 77
.:64B8 AA AA AA AA A8 80 95 55
.:64C0 FC F0 C0 00 02 2A AA AA
.:64C8 0F 03 00 A8 AA AA AA AA
.:64D0 AA AA AA 2A FE FF FF FF
.:64D8 54 54 50 50 43 BF BF BF
.:64E0 00 00 00 00 80 A2 AA AA
.:64E8 3F 0F 0F 00 A8 AA AA AA
.:64F0 FF FF FF FF 3F BF BF AF
.:64F8 FF FF FF FF FF FF FF FF
.:6500 AF AF A9 A9 AA AA 55 55
.:6508 FE FA AA AA 6A 9A 55 55
.:6510 55 55 55 55 55 55 AA AA
.:6518 AA AA AA AA AA AA FF FF
.:6520 56 59 59 59 65 95 AA AA
.:6528 AA AA AA AA AA AA FF FF
.:6530 A2 A2 AA AA AA AA FF FF
.:6538 AA AA AA AA AA AA 55 55
.:6540 AA AA AA AA AA AA 55 55
.:6548 AA AA AA AA AA AA 55 55
.:6550 7D 59 59 56 55 55 AA AA
.:6558 AA AA AA AA 6A 9A 55 55
.:6560 55 55 55 55 55 55 AA AA
.:6568 A9 AA AA AA AA AA FF FF
.:6570 AA AA AA 6A 59 55 FF FF
.:6578 AF BF FF FF FF FF 55 55
.:6580 03 A9 AA AA A9 A6 55 55
.:6588 DA 6A 6A 6A AA AA 55 55
.:6590 AA AA AA AA AA AA 55 55
.:6598 AA AA AA AA AA AA FF FF
.:65A0 A9 AA AA AA AA AA 55 55
.:65A8 AA 5A A5 A6 9A 6A 55 55
```

```
.:65B0 9A 6A AA AA AA AA 55 55
.:65B8 AA AA AA AA AA AA 55 55
.:65C0 55 55 55 55 55 57 AA AA
.:65C8 AA AA AB AE BA EA 55 55
.:65D0 75 D5 55 55 55 55 AA AA
.:65D8 1D 47 1D 47 1D 47 1D 47
.:65E0 95 66 99 66 99 66 99 66
.:65E8 AE B9 EE B9 EE B9 EE B9
.:65F0 EE BB EE BB EE BB EE BB
.:65F8 AA AA AA AA AA AA FF FF
.:6600 AA AA AA AA AA AA 55 55
.:6608 AA AA AA AA AA AA 55 55
.:6610 A9 A9 A6 A6 9A 6A 55 55
.:6618 FF FF FF FF FF FF 55 55
.:6620 AA AA AA AA AA AA 55 55
.:6628 55 55 55 55 55 55 AA AA
.:6630 AB AA AA AA AA AA 55 55
.:6638 FF FE 9A 9A A6 A6 55 55
.:6640 AA AA AA AA AA AA AA AA
.:6648 AA AA AA AA AA AA AA AA
.:6650 AA AA AA AA AA AA AA AA
.:6658 AA AA AA AA AA AA AA AA
.:6660 AA AA AA AA AA AA AA AA
.:6668 AA AA AA AA AA AA AA AA
.:6670 AA AA AA AA AA AA AA AA
.:6678 AA AA AA AA AA AA AA AA
.:6680 AA AA AA AA AA AA AA AA
.:6688 FF FF FF FF FF FF FF FF
.:6690 AA AA AA AA AA AA AA AA
.:6698 AA AA AA AA AA AA AA AA
.:66A0 AA AA AA AA AA AA AA AA
.:66A8 FF FF FF FF FF FF FF FF
.:66B0 55 55 55 55 55 55 55 55
.:66B8 FF FF FF FF FF FF FF FF
.:66C0 FF FF FF FF FF FF FF FF
.:66C8 AA AA AA AA AA AA AA AA
.:66D0 AA AA AA AA AA AA AA AA
.:66D8 AA AA AA AA AA AA AA AA
.:66E0 55 55 55 55 55 55 55 55
.:66E8 AA AA AA AA AA AA AA AA
.:66F0 AA AA AA AA AA AA AA AA
.:66F8 55 55 55 55 55 55 55 55
.:6700 AA AA AA AA AA AA AA AA
.:6708 AA AA AA AA AA AA AA AA
.:6710 AA AA AA AA AA AA AA AA
.:6718 26 89 26 89 26 89 26 89
.:6720 99 66 99 66 99 66 99 66
.:6728 77 DE 77 DE 77 DE 77 DE
.:6730 DD 77 DD 77 DD 77 DD 77
.:6738 55 55 55 BB EE BB EE BB
.:6740 AA AA AA 77 DD 77 DD 77
.:6748 AA AA AB DD 77 DD 77 DD
.:6750 AA AA 77 DD 77 DD 77 DD
.:6758 FF FF 66 99 66 99 66 99
.:6760 AA AA 77 DD 77 DD 77 DD
.:6768 FF F9 66 99 66 99 66 99
.:6770 AA DD 77 DD 77 DD 77 DD
.:6778 55 EE BB EE BB EE BB EE
.:6780 AA AA AA AA AA AA AA AA
.:6788 FF FF FF FF FF FF FF FF
.:6790 AA AA AA AA AA AA AA AA
.:6798 FF FF FF FF FF FF FF FF
.:67A0 AA AA AA AA AA AA AA AA
.:67A8 AA AA AA AA AA AA AA AA
.:67B0 AA AA AA AA AA AA AA AA
.:67B8 FF FF FF FF FF FF FF FF
.:67C0 FF FF FF FF FF FF FF FF
.:67C8 AA AA AA AA AA AA AA AA
.:67D0 FF FF FF FF FF FF FF FF
.:67D8 AA AA AA AA AA AA AA AA
.:67E0 AA AA AA AA AA AA AA AA
.:67E8 AA AA AA AA AA AA AA AA
.:67F0 AA AA AA AA AA AA AA AA
.:67F8 AA AA AA AA AA AA AA AA
.:6800 AA AA AA AA AA AA AA AA
.:6808 AA AA AA AA AA AA AA AA
.:6810 AA AA AA AA AA AA AA AA
.:6818 AA AA AA AA AA AA AA AA
.:6820 AA AA AA AA AA AA AA AA
.:6828 AA AA AA AA AA AA AA AA
.:6830 AA AA AA AA AA AA AA AA
.:6838 AA AA AA AA AA AA AA AA
.:6840 AA AA AA AA AA AA AA AA
.:6848 55 55 55 56 56 56 5B 6F
.:6850 FA E5 95 55 55 55 55 55
.:6858 37 CD 37 CD 37 CD 37 CD
.:6860 66 99 66 99 66 99 66 99
.:6868 77 DE 77 DE 77 DE 77 DE
.:6870 EE BB EE BB EE BB EE BB
.:6878 99 66 AE 7B AE 6B 9E 67
.:6880 BB EE 66 99 66 99 66 99
.:6888 BB 99 66 99 66 99 66 99
.:6890 99 BB EE BB EE BB EE BB
.:6898 E7 DD 77 DD 77 DD 77 DD
.:68A0 EE BB EE BB EE BB EE BB
.:68A8 BB EE BB EE BB EE BB EE
.:68B0 EE BB EE BB EE BB EE BB
.:68B8 66 99 66 99 66 99 66 99
.:68C0 FF FF FF FF FF FF FF FF
.:68C8 AA AA AA AA AA AA AA AA
.:68D0 AA AA AA AA AA AA AA AA
.:68D8 FF FF FF FF FF FF FF FF
.:68E0 FF FF FF FF FF FF FF FF
.:68E8 FF FF FF FF FF FF FF FF
.:68F0 FF FF FF FF FF FF FF FF
.:68F8 FF FF FF FF FF FF FF FF
.:6900 FF FF FF FF FF FF FF FF
.:6908 FF FF FF FF FF FF FF FF
.:6910 AA AA AA AA AA AA AA AA
.:6918 AA AA AA AA AA AA AA AA
.:6920 FF FF FF FF FF FF FF FF
.:6928 FF FF FF FF FF FF FF FF
.:6930 AA AA AA AA AA AA AA AA
.:6938 AA AA AA AA AA AA AA AA
.:6940 AA AA AA AA AA AA AA AA
.:6948 AA AA AA AA AA AA AA AA
.:6950 FF FF FF FF FF FF FF FF
.:6958 FF FF FF FF FF FF FF FF
```

```
.:6960 AA AA AA AA AA AA AA AA
.:6968 FF FF FF FF FF FF FF FF
.:6970 FF FF FF FF FF FF FF FF
.:6978 FF FF FF FF FF FF FF FF
.:6980 AA AA AA AA AA AB AB AF
.:6988 7A 7A EA EA EA AA AA AA
.:6990 FF FF FF FF FF FF FF FF
.:6998 1D 47 1D 47 1D 47 1D 47
.:69A0 99 66 99 66 99 66 99 66
.:69A8 BB DE B7 DE BB DD B7 DD
.:69B0 99 66 99 66 99 66 D9 66
.:69B8 BA ED BB EE BB EE BB EE
.:69C0 99 66 99 66 99 E6 59 D6
.:69C8 BB EE BB EE BB EE BB EE
.:69D0 BB EE BB EE BB EE BB EE
.:69D8 BB EE BB EE BB EE BB EE
.:69E0 BB EE BB EE BB EE BB EE
.:69E8 BB EE BB EE BB EE BB EE
.:69F0 BB EE BB EE BB EE BB EE
.:69F8 EE BB EE BB EE BB EE BB
.:6A00 AA AA AA AA AA AA AA AA
.:6A08 AA AA AA AA AA AA AA AA
.:6A10 AA AA AA AA AA AA AA AA
.:6A18 FF FF FF FF FF FF FF FF
.:6A20 FF FF FF FF FF FF FF FF
.:6A28 FF FF FF FF FF FF FF FF
.:6A30 FF FF FF FF FF FF FF FF
.:6A38 55 55 55 55 55 55 55 55
.:6A40 AA AA AA AA AA AA AA AA
.:6A48 FF FF FF FF FF FF FF FF
.:6A50 FF FF FF FF FF FF FF FF
.:6A58 FF FF FF FF FF FF FF FF
.:6A60 FF FF FF FF FF FF FF FF
.:6A68 FF FF FF FF FF FF FF FF
.:6A70 55 55 55 55 55 55 55 55
.:6A78 FF FF FF FF FF FF FF FF
.:6A80 FF FF FF FF FF FF FF FF
.:6A88 AA AA AA AA AA AA AA AA
.:6A90 AA AA AA AA AA AA AA AA
.:6A98 AA AA AA AA AA AA AA AA
.:6AA0 AA AA AA AA AA AA AA AA
.:6AA8 FF FF FF FF FF FF FF FF
.:6AB0 FF FF FF FF FF FF FF FF
.:6AB8 FF FF FF FF FF FF FF FF
.:6AC0 5B 5B 6F 6F 6F 6F 6F 6F
.:6AC8 AA AA AA AA AA AA AA AA
.:6AD0 AA AA AA AA AA AA AA AA
.:6AD8 2E 8B 2E 8B 2E 8B 2E 8B
.:6AE0 BB EE BB EE BB EE BB EE
.:6AE8 6E BB 6E BB 6E BB 6E BB
.:6AF0 6E 9B 66 9B 66 99 66 99
.:6AF8 EE BB EE BB EE BB 6E BB
.:6B00 DB 7E DF 76 DD 77 DD 77
.:6B08 BB EE BB EE BB EE BB 6E
.:6B10 BB EE BB EE 95 D5 95 D5
.:6B18 DD 77 DD 77 DA 7A DA 7A
.:6B20 EE BB EE BB ED B9 ED B9
.:6B28 DD 77 DD 77 9D B7 9D B7
.:6B30 66 99 66 99 F6 F9 F6 F9
.:6B38 EE BB EE BB D6 97 D6 97
.:6B40 AA AA AA AA AA AA AA AA
.:6B48 55 55 55 55 55 55 55 55
.:6B50 AA AA AA AA AA AA AA AA
.:6B58 AA AA AA AA AA AA AA AA
.:6B60 AA AA AA AA AA AA AA AA
.:6B68 AA AA AA AA AA AA AA AA
.:6B70 FF FF FF FF FF FF FF FF
.:6B78 FF FF FF FF FF FF FF FF
.:6B80 FF FF FF FF FF FF FF FF
.:6B88 FF FF FF FF FF FF FF FF
.:6B90 FF FF FF FF FF FF FF FF
.:6B98 FF FF FF FF FF FF FF FF
.:6BA0 AA AA AA AA AA AA AA AA
.:6BA8 AA AA AA AA AA AA AA AA
.:6BB0 AA AA AA AA AA AA AA AA
.:6BB8 AA AA AA AA AA AA AA AA
.:6BC0 55 55 55 55 55 55 55 55
.:6BC8 FF FF FF FF FF FF FF FF
.:6BD0 FF FF FF FF FF FF FF FF
.:6BD8 FF FF FF FF FF FF FF FF
.:6BE0 FF FF FF FF FF FF FF FF
.:6BE8 55 55 55 55 55 55 55 55
.:6BF0 AA AA AA AA AA AA AA AA
.:6BF8 AA AA AB AB AD AD B5 B5
.:6C00 95 95 55 55 55 55 55 55
.:6C08 AA AA AA AA AA AA AA AA
.:6C10 FF FF FF FF FF FF FF FF
.:6C18 3B CE 3B CE 3B CE 3B CE
.:6C20 DD 77 DD 77 DD 77 DD 77
.:6C28 7B EE 7B EE 7B EE 7B EE
.:6C30 99 66 99 66 99 66 99 66
.:6C38 E6 B9 FE B9 EE BB EE BB
.:6C40 66 99 66 99 66 99 E6 99
.:6C48 5D B7 6D 9B 66 9B 66 99
.:6C50 D5 95 D5 95 ED B7 DD 77
.:6C58 5A 5A AA AA EE BB EE BB
.:6C60 DE 76 AA AA EE BB EE BB
.:6C68 E6 D9 FF FF 77 DD 77 DD
.:6C70 F6 F9 FF EE 7B EE BB EE
.:6C78 D6 97 56 65 99 65 99 65
.:6C80 FF FF FF FF FF FF FF FF
.:6C88 FF FF FF FF FF FF FF FF
.:6C90 FF FF FF FF FF FF FF FF
.:6C98 FF FF FF FF FF FF FF FF
.:6CA0 FF FF FF FF FF FF FF FF
.:6CA8 FF FF FF FF FF FF FF FF
.:6CB0 FF FF FF FF FF FF FF FF
.:6CB8 55 55 55 55 55 55 55 55
.:6CC0 55 55 55 55 55 55 55 55
.:6CC8 55 55 55 55 55 55 55 55
.:6CD0 55 55 55 55 55 55 55 55
.:6CD8 FF FF FF FF FF FF FF FF
.:6CE0 FF FF FF FF FF FF FF FF
.:6CE8 FF FF FF FF FF FF FF FF
.:6CF0 FF FF FF FF FF FF FF FF
.:6CF8 FF FF FF FF FF FF FF FF
.:6D00 AA AA AA AA AA AA AA AA
.:6D08 55 55 55 55 55 55 55 55
```

```
.:6D10 55 55 55 55 55 55 55 55
.:6D18 55 55 55 55 55 55 55 55
.:6D20 AA AA AA AA AA AA AA AA
.:6D28 55 55 55 55 55 55 55 55
.:6D30 FF FF FD FD FD FD FD FF
.:6D38 EA EA AA AA AA AA AA EA
.:6D40 FF FF FF FF FF FF FF FF
.:6D48 FF FF FF FF FF FF FF FF
.:6D50 FF FF FF FF FF FF FF FF
.:6D58 3B CE 3B CE 3B CE 3B CE
.:6D60 77 DD 77 DD 77 DD 77 DD
.:6D68 7B EE 7B EE 7B EE 7B EE
.:6D70 99 66 99 66 95 65 95 65
.:6D78 BB EE BB EE BB EE BB EE
.:6D80 9D 67 9D 67 99 66 99 66
.:6D88 BB EE BB EE BB EE BB EE
.:6D90 D9 66 D9 76 DD 77 DD 77
.:6D98 77 DD 77 DD 77 DD 37 CD
.:6DA0 9D 77 9D 77 9D 77 9D 55
.:6DA8 BB EE BB EE BB EE BB FF
.:6DB0 99 66 99 66 99 66 99 66
.:6DB8 BB EF BB EF BB EE BB EE
.:6DC0 FF FF FF FF FF FF FF FF
.:6DC8 FF FF FF FF FF FF FF FF
.:6DD0 FF FF FF FF FF FF FF FF
.:6DD8 FF FF FF FF FF FF FF FF
.:6DE0 AA AA AA AA AA AA AA AA
.:6DE8 55 55 55 55 55 55 55 55
.:6DF0 AA AA AA AA AA AA AA AA
.:6DF8 AA AA AA AA AA AA AA AA
.:6E00 AA AA AA AA AA AA AA AA
.:6E08 AA AA AA AA AA AA AA AA
.:6E10 AA AA AA AA AA AA AA AA
.:6E18 AA AA AA AA AA AA AA AA
.:6E20 FF FF FF FF FF FF FF FF
.:6E28 AA AA AA AA AA AA AA AA
.:6E30 FF FF FF FF FF FF FF FF
.:6E38 FF FF FF FF FF FF FF FF
.:6E40 AA AA AA AA AA AA AA AA
.:6E48 FF FF FF FF FF FF FF FF
.:6E50 AA AA AA AA AA AA AA AA
.:6E58 AA AA AA AA AA AA AA AA
.:6E60 AA AA AA AA AA AA AA AA
.:6E68 55 55 55 55 55 55 55 55
.:6E70 FF FF FF FF FF FF FF FF
.:6E78 7F 7F 7F 7F 7F 7F 7F 7F
.:6E80 FF FF FF FF FF FF FF FF
.:6E88 FF FF FF FF FF FF FF FF
.:6E90 FF FF FF FF FF FF FF FF
.:6E98 2E 8B 2E 8B 2E 8B 2E 8B
.:6EA0 99 66 99 66 99 66 99 66
.:6EA8 9D 77 9D 77 9D 77 9D 77
.:6EB0 95 65 95 65 99 65 99 66
.:6EB8 FB FE FB FE FF FE FF FE
.:6EC0 EE BB EE BB EE BB EE BB
.:6EC8 E6 B9 E6 B9 EE BB EE BB
.:6ED0 EE BB EE BB EE BB 6E BB
.:6ED8 37 CD 37 CD 37 CD 37 CD
.:6EE0 BF FF BF FF BF FF BF FF
```

```
.:6EE8 FF FF FF FF FF FF FF FF
.:6EF0 BB EE BB EE BB EE BB EE
.:6EF8 BB EE BB EE BB EE BB EE
.:6F00 FF FF FF FF FF FF FF FF
.:6F08 FF FF FF FF FF FF FF FF
.:6F10 FF FF FF FF FF FF FF FF
.:6F18 55 55 55 55 55 55 55 55
.:6F20 AA AA AA AA AA AA AA AA
.:6F28 AA AA AA AA AA AA AA AA
.:6F30 AA AA AA AA AA AA AA AA
.:6F38 AA AA AA AA AA AA AA AA
.:6F40 AA AA AA AA AA AA AA AA
.:6F48 55 55 55 55 55 55 55 55
.:6F50 FF FF FF FF FF FF FF FF
.:6F58 FF FF FF FF FF FF FF FF
.:6F60 FF FF FF FF FF FF FF FF
.:6F68 FF FF FF FF FF FF FF FF
.:6F70 FF FF FF FF FF FF FF FF
.:6F78 FF FF FF FF FF FF FF FF
.:6F80 AA AA AA AA AA AA AA AA
.:6F88 AA AA AA AA AA AA AA AA
.:6F90 AA AA AA AA AA AA AA AA
.:6F98 AA AA AA AA AA AA AA AA
.:6FA0 AA AA AA AA AA AA AA AA
.:6FA8 AA AA AA AA AA AA AA AA
.:6FB0 AB AB AB AB AB AB AD AD
.:6FB8 55 55 55 55 55 55 55 55
.:6FC0 FF FF FF FF FF FF FF FF
.:6FC8 FF FF FF FF FF FF FF FF
.:6FD0 FF FF FF FF FF FF FF FF
.:6FD8 26 89 26 89 26 89 26 89
.:6FE0 99 66 99 66 99 66 99 66
.:6FE8 6E BB 6E BB 6E BB 6E BB
.:6FF0 DD 77 DD 77 DD 77 DD 77
.:6FF8 EA AB EA BB EE BB EE BB
.:7000 EE BB EE BB EE BB EE BB
.:7008 EE BB EE BB EE BB EA BB
.:7010 E6 B9 EE B9 EE BB EE BB
.:7018 3B CE 3B CE 3B CE 3B CE
.:7020 B7 DD B7 DD B7 DD B7 DD
.:7028 77 DD 77 DD 77 DD 77 DD
.:7030 EE BB EE BB EE BB EE BB
.:7038 BB EE BB EE BB EE BB DD
.:7040 55 55 55 55 55 55 55 55
.:7048 55 55 55 55 55 55 55 55
.:7050 FF FF FF FF FF FF FF FF
.:7058 55 55 55 55 55 55 55 55
.:7060 55 55 55 55 55 55 55 55
.:7068 55 55 55 55 55 55 55 55
.:7070 55 55 55 55 55 55 55 55
.:7078 55 55 55 55 55 55 55 55
.:7080 55 55 55 55 55 55 55 55
.:7088 55 55 55 55 55 55 55 55
.:7090 AA AA AA AA AA AA AA AA
.:7098 55 55 55 55 55 55 55 55
.:70A0 55 55 55 55 55 55 55 55
.:70A8 55 55 55 55 55 55 55 55
.:70B0 AA AA AA AA AA AA AA AA
.:70B8 FF FF FF FF FF FF FF FF
```

```
.:70C0 55 55 55 55 55 55 55 55
.:70C8 55 55 55 55 55 55 55 55
.:70D0 55 55 55 55 55 55 55 55
.:70D8 55 55 55 55 55 55 55 55
.:70E0 55 55 55 55 55 55 55 55
.:70E8 55 55 55 55 55 55 55 55
.:70F0 F9 F9 F9 F9 F9 F9 F9 E5
.:70F8 FF FF FF FF FF FF FF FF
.:7100 FF FF FF FF FF FF FF FF
.:7108 FF FF FF FF FF FF FF FF
.:7110 FF FF FF FF FF FF FF FF
.:7118 3B CE 3B CE 3B CE 3B CE
.:7120 99 66 99 66 99 66 99 66
.:7128 7B EE 7B EE 7B EE 7B EE
.:7130 99 66 99 66 99 66 99 66
.:7138 BB EE BB EE BB EE BB EE
.:7140 BB EE BB EE BB EE BB EE
.:7148 BF EF BF EF BF EF BF EF
.:7150 BB EE FB EE FB FE FB FE
.:7158 3B CE 3B CE 3B CE 3B CE
.:7160 7B EE 7B EE 7B EE 7B EE
.:7168 77 DD 77 DD 77 DD 77 DD
.:7170 BB EE BB EE BB EE BB EE
.:7178 B7 DD B7 DD B7 DD B7 DD
.:7180 AA AA AA AA AA AA AA AA
.:7188 55 55 55 55 55 55 55 55
.:7190 FF FF FF FF FF FF FF FF
.:7198 FF FF FF FF FF FF FF FF
.:71A0 FF FF FF FF FF FF FF FF
.:71A8 FF FF FF FF FF FF FF FF
.:71B0 FF FF FF FF FF FF FF FF
.:71B8 FF FF FF FF FF FF FF FF
.:71C0 FF FF FF FF FF FF FF FF
.:71C8 AA AA AA AA AA AA AA AA
.:71D0 AA AA AA AA AA AA AA AA
.:71D8 AA AA AA AA AA AA AA AA
.:71E0 AA AA AA AA AA AA AA AA
.:71E8 AA AA AA AA AA AA AA AA
.:71F0 AA AA AA AA AA AA AA AA
.:71F8 55 55 55 55 55 55 55 55
.:7200 FF FF FF FF FF FF FF FF
.:7208 FF FF FF FF FF FF FF FF
.:7210 FF FF FF FF FF FF FF FF
.:7218 FF FF FF FF FF FF FF FF
.:7220 FF FF FF FF FF FF FF FF
.:7228 AA AA AA AA AA AA AA AA
.:7230 9F 9F 9F 9F 9F 9F 9F 9F
.:7238 AA AA AA AA AA AA AA AA
.:7240 AA AA AA AA AA AA AA AA
.:7248 AA AA AA AA AA AA AA AA
.:7250 AA AA AA AA AA AA AA AA
.:7258 2E 8B 2E 8B 2E 8B 2E 8B
.:7260 66 99 66 99 66 99 66 99
.:7268 6E BB 6E BB 6E BB 6E BB
.:7270 99 66 99 66 99 66 99 66
.:7278 99 66 99 66 99 66 99 66
.:7280 99 66 99 66 99 66 99 66
.:7288 77 DF 77 DD 77 DD 77 DD
.:7290 59 56 59 56 99 66 99 66
.:7298 37 CD 37 CD 37 CD 37 CD
.:72A0 9D 77 9D 77 9D 77 9D 77
.:72A8 66 99 66 99 66 99 66 99
.:72B0 DD 77 DD 77 DD 77 DD 77
.:72B8 6E BB 6E BB 6E BB 6E BB
.:72C0 55 55 55 55 55 55 55 55
.:72C8 FF FF FF FF FF FF FF FF
.:72D0 FF FF FF FF FF FF FF FF
.:72D8 FF FF FF FF FF FF FF FF
.:72E0 FF FF FF FF FF FF FF FF
.:72E8 FF FF FF FF FF FF FF FF
.:72F0 FF FF FF FF FF FF FF FF
.:72F8 FF FF FF FF FF FF FF FF
.:7300 AA AA AA AA AA AA AA AA
.:7308 AA AA AA AA AA AA AA AA
.:7310 AA AA AA AA AA AA AA AA
.:7318 AA AA AA AA AA AA AA AA
.:7320 AA AA AA AA AA AA AA AA
.:7328 AA AA AA AA AA AA AA AA
.:7330 FF FF FF FF FF FF FF FF
.:7338 FF FF FF FF FF FF FF FF
.:7340 FF FF FF FF FF FF FF FF
.:7348 FF FF FF FF FF FF FF FF
.:7350 FF FF FF FF FF FF FF FF
.:7358 FF FF FF FF FF FF FF FF
.:7360 FF FF FF FF FF FF FF FF
.:7368 FF FF FF FF FF FF FF FF
.:7370 DA DA DA DA DA DA DA DA
.:7378 AA AA AA AA AA AA AA AA
.:7380 AA AA AA AA AA AA AA AA
.:7388 AA AA AA AA AA AA AA AA
.:7390 FF FF FF FF FF FF FF FF
.:7398 37 CD 37 CD 37 CD 37 CD
.:73A0 BB EE BB EE BB EE BB EE
.:73A8 6E BB 6E BB 6E BB 6E BB
.:73B0 99 66 99 66 99 66 99 65
.:73B8 BB EE BB FF FF FF FF FF
.:73C0 99 66 99 66 59 66 59 56
.:73C8 99 66 99 66 99 66 99 66
.:73D0 99 66 99 66 99 66 99 66
.:73D8 3B CE 3B CE 3B CE 3B CE
.:73E0 9D 77 9D 77 9D 77 9D 77
.:73E8 BB EE BB EE BB EE BB EE
.:73F0 66 99 66 99 66 99 66 99
.:73F8 E6 99 E6 99 E6 99 E6 99
.:7400 FF FF FF FF FF FF FF FF
.:7408 FF FF FF FF FF FF FF FF
.:7410 FF FF FF FF FF FF FF FF
.:7418 FF FF FF FF FF FF FF FF
.:7420 FF FF FF FF FF FF FF FF
.:7428 FF FF FF FF FF FF FF FF
.:7430 FF FF FF FF FF FF FF FF
.:7438 55 55 55 55 55 55 55 55
.:7440 AA AA AA AA AA AA AA AA
.:7448 AA AA AA AA AA AA AA AA
.:7450 AA AA AA AA AA AA AA AA
.:7458 AA AA AA AA AA AA AA AA
.:7460 AA AA AA AA AA AA AA AA
.:7468 AA AA AA AA AA AA AA AA
```

```
.:7470 55 55 55 55 55 55 55 55
.:7478 FF FF FF FF FF FF FF FF
.:7480 AA AA AA AA AA AA AA AA
.:7488 FF FF FF FF FF FF FF FF
.:7490 FF FF FF FF FF FF FF FF
.:7498 FF FF FF FF FF FF FF FF
.:74A0 FF FF FF FF FF FF FF FF
.:74A8 FF FF FF FF FF FF FF FF
.:74B0 97 A7 A7 A7 A5 A9 A9 A9
.:74B8 55 55 55 55 55 55 55 D5
.:74C0 AA AA AA AA AA AA AA AA
.:74C8 FF FF FF FF FF FF FF FF
.:74C0 AA AA AA AA AA AA AA AA
.:74C8 FF FF FF FF FF FF FF FF
.:74D0 FF FF FF FF FF FF FF FF
.:74D8 37 CD 37 CD 37 CD 37 9D
.:74E0 BB EE BB EE BB EE BB EE
.:74E8 6E BB 6E BB 6E BB 6E BB
.:74F0 EE BA EA BA EA BA EA BA
.:74F8 AA AA AA AA AA AA AA AA
.:7500 59 56 55 56 55 56 55 56
.:7508 99 66 99 66 99 66 99 66
.:7510 99 66 99 66 99 66 99 66
.:7518 3B CE 3B CE 3B CE 3B CE
.:7520 B7 DD B7 DD B7 DD B7 DD
.:7528 BB EE BB EE BB EE BB EE
.:7530 BB EE BB EE BB EE BB EE
.:7538 6E BB 6E BB 6E BB 6E BB
.:7540 FF FF FF FF FF FF FF FF
.:7548 FF FF FF FF FF FF FF FF
.:7550 FF FF FF FF FF FF FF FF
.:7558 FF FF FF FF FF FF FF FF
.:7560 FF FF FF FF FF FF FF FF
.:7568 FF FF FF FF FF FF FF FF
.:7570 FF FF FF FF FF FF FF FF
.:7578 55 55 55 55 55 55 55 55
.:7580 55 55 55 55 55 55 55 55
.:7588 55 55 55 55 55 55 55 55
.:7590 AA AA AA AA AA AA AA AA
.:7598 55 55 55 55 55 55 55 55
.:75A0 AA AA AA AA AA AA AA AA
.:75A8 55 55 55 55 55 55 55 55
.:75B0 AA AA AA AA AA AA AA AA
.:75B8 FF FF FF FF FF FF FF FF
.:75C0 FF FF FF FF FF FF FF FF
.:75C8 FF FF FF FF FF FF FF FF
.:75D0 FF FF FF FF FF FF FF FF
.:75D8 FF FF FF FF FF FF FF FF
.:75E0 FF FF FF FF FF FF FF FF
.:75E8 FF FF FF FF FF FF FF FF
.:75F0 FF FF FF FF FF FF FF FF
.:75F8 BF BF BF BF BF BF BF BF
.:7600 AA AA AA AA AA AA AA AA
.:7608 AA AA AA AA AA AA AA AA
.:7610 AA AA AA AA AA AA AA AA
.:7618 5E 57 55 55 55 55 55 55
.:7620 99 66 7F 7F 7F 7F 5F 5F
.:7628 6B A8 AB AA AA AA AA AA
.:7630 D5 75 D5 75 D5 B5 95 B5
.:7638 AA AA AA AA AA AA AA AA
.:7640 AA A9 AA A9 AA A9 AA A9
.:7648 99 66 99 66 99 66 99 66
.:7650 66 99 66 99 66 99 66 99
.:7658 26 89 26 89 26 89 26 89
.:7660 6E BB 6E BB 6E BB 6E BB
.:7668 EE BB EE BB EE BB EE BB
.:7670 66 99 66 99 66 99 66 99
.:7678 9D 77 9D 77 9D 77 9D 77
.:7680 AA AA AA AA FF 77 DD 77
.:7688 AA AA AA AA FF 77 DD 77
.:7690 AA AA AA AA FF 77 DD 77
.:7698 AA AA AA AA FF 77 DD 77
.:76A0 AA AA AA AA 55 DD 77 DD
.:76A8 55 55 55 55 AA EE BB EE
.:76B0 FF FF FF FF AA 66 99 66
.:76B8 FF FF FF FF AA 66 99 66
.:76C0 FF FF FF FF AA 66 99 66
.:76C8 FF FF FF FF AA 66 99 66
.:76D0 FF FF FF FF 55 99 66 99
.:76D8 FF FF FF FF AA 66 99 66
.:76E0 FF FF FF FF 55 99 66 99
.:76E8 FF FF FF FF 55 99 66 99
.:76F0 55 55 55 55 FF BB EE BB
.:76F8 AA AA AA AA FF 77 DD 77
.:7700 AA AA AA AA FF 77 DD 77
.:7708 AA AA AA AA FF 77 DD 77
.:7710 AA AA AA AA FF 77 DD 77
.:7718 AA AA AA AA FF 77 DD 77
.:7720 AA AA AA AA FF DD 77 DD
.:7728 AA AA AA AA FF DD 77 DD
.:7730 AA AA AA AA FF DD 77 DD
.:7738 B5 B5 B5 B5 FD ED AD AB
.:7740 FF FF FF FF FF FF FF FF
.:7748 55 55 55 55 55 55 55 55
.:7750 FF FF FF FF FF FF FF FF
.:7758 FF FF FF FF FF FF FF FF
.:7760 7E 5F 57 57 57 57 55 55
.:7768 AA AA AA AA AA EA EA EA
.:7770 AF AF AF AB AB AA AA AA
.:7778 FF FF FF FF FF FF FF 7F
.:7780 55 56 55 56 55 56 55 56
.:7788 DD 77 DD 77 DD 77 DD 77
.:7790 DD 77 DD 77 DD 77 DD 77
.:7798 37 CD 37 CD 37 CD 37 CD
.:77A0 E6 99 E6 99 E6 99 E6 99
.:77A8 66 99 66 99 66 99 66 99
.:77B0 BB EE BB EE BB EE BB EE
.:77B8 B7 DD B7 DD B7 DD B7 DD
.:77C0 DD 77 DD 77 DD 77 DD 77
.:77C8 DD 77 DD 77 DD 77 DD 77
.:77D0 DD 77 DD 77 DD 77 DD 77
.:77D8 DD 77 DD 77 DD 77 DD 77
.:77E0 DD 77 DD 77 DD 77 DD 77
.:77E8 99 66 99 66 99 66 99 66
.:77F0 66 99 66 99 66 99 66 99
.:77F8 66 99 66 99 66 99 66 99
.:7800 66 99 66 99 66 99 66 99
.:7808 99 66 99 66 99 66 99 66
```

```
.:7810 77 DD 77 DD 77 DD 77 DD
.:7818 99 66 99 66 99 66 99 66
.:7820 99 66 99 66 99 66 99 66
.:7828 EE BB EE BB EE BB EE BB
.:7830 EE BB EE BB EE BB EE BB
.:7838 DD 77 DD 77 DD 77 DD 77
.:7840 BB EE BB EE BB EE BB EE
.:7848 77 DD 77 DD 77 DD 77 DD
.:7850 77 DD 77 DD 77 DD 77 DD
.:7858 77 DD 77 DD 77 DD 77 DD
.:7860 77 DD 77 DD 77 DD 77 DD
.:7868 77 DD 77 DD 77 DD 77 DD
.:7870 DD 77 DD 77 DD 77 DD 77
.:7878 EF BB EE BB EE BB EE BB
.:7880 FF BF BF BF BF 6F 6F 5E
.:7888 FF FF FF FF FF FF FF FF
.:7890 FF FF FF FF FF FF FF FF
.:7898 FF FF FF FF FF FF FF FF
.:78A0 FF FF FF FF FF FF FF FF
.:78A8 D5 DF D5 F5 B5 AD AD AD
.:78B0 AA AA AA AA AA AA AA AA
.:78B8 7F 5F 5F 5F 5F 57 57 57
.:78C0 FF FE FF FE FF FE FF FE
.:78C8 99 66 99 66 99 66 99 66
.:78D0 DD 77 DD 77 DD 77 DD 77
.:78D8 37 CD 37 CD 37 CD 37 CD
.:78E0 E6 99 E6 99 E6 99 E6 99
.:78E8 EE BB EE BB EE BE EE BB
.:78F0 EE BB EE BB EE BB EE BB
.:78F8 E6 99 E6 99 E6 99 E6 99
.:7900 DD 77 DD 77 DD 77 DD 77
.:7908 DD 77 DD 77 DD 77 DD 77
.:7910 77 DD 77 DD 77 DD 77 DD
.:7918 DD 77 DD 77 DD 77 DD 77
.:7920 DD 77 DD 77 DD 77 DD 77
.:7928 99 66 99 66 99 66 99 66
.:7930 66 99 66 99 66 99 66 99
.:7938 99 66 99 66 99 66 99 66
.:7940 EE BB EE BB EE BB EE BB
.:7948 99 66 99 66 99 66 99 66
.:7950 DD 77 DD 77 DD 77 DD 77
.:7958 99 66 99 66 99 66 99 66
.:7960 77 DD 77 DD 77 DD 77 DD
.:7968 99 66 99 66 99 66 99 66
.:7970 EE BB EE BB EE BB EE BB
.:7978 DD 77 DD 77 DD 77 DD 77
.:7980 DD 77 DD 77 DD 77 DD 77
.:7988 77 DD 77 DD 77 DD 77 DD
.:7990 77 DD 77 DD 77 DD 77 DD
.:7998 77 DD 77 DD 77 DD 77 DD
.:79A0 DD 77 DD 77 DD 77 DD 77
.:79A8 DD 77 DD 77 DD 77 DD 77
.:79B0 77 DD 77 DD 77 DD 77 DD
.:79B8 77 DD 77 DD 77 DD 77 DD
.:79C0 AE BB EF BB EE BB EE BB
.:79C8 55 55 55 55 D5 D5 D5 D5
.:79D0 FF FF FF FF FF FF FF FF
.:79D8 FF FF FF FF FF FF FF FF
.:79E0 FF FF FF FF FF FF FF FF
.:79E8 F9 F9 FE FE FE FE FE FE
.:79F0 AA AA AA AA AA AA AA EA
.:79F8 AB AB AA AA AA AA AA AA
.:7A00 55 56 55 96 A5 A6 AD A7
.:7A08 66 99 66 99 66 99 66 99
.:7A10 66 99 66 99 66 99 66 99
.:7A18 3B CE 3B CE 3B CE 3B CE
.:7A20 E6 99 E6 99 E6 99 E6 99
.:7A28 77 DD 77 DD 77 DD 77 DD
.:7A30 EE BB EE BB EE BB EE BB
.:7A38 E6 99 E6 99 E6 99 E6 99
.:7A40 DD 77 DD 77 DD FF AA AA
.:7A48 DD 77 DD 77 DD FF AA AA
.:7A50 77 DD 77 DD 77 55 AA AA
.:7A58 EE BB EE BB EE FF 55 55
.:7A60 99 66 99 66 99 AA FF FF
.:7A68 99 66 99 66 99 AA FF FF
.:7A70 66 99 66 99 66 55 FF FF
.:7A78 99 66 99 66 99 AA FF FF
.:7A80 EE BB EE BB EE FF 55 55
.:7A88 99 66 99 66 99 AA FF FF
.:7A90 99 66 99 66 99 AA FF FF
.:7A98 99 66 99 66 99 AA FF FF
.:7AA0 99 66 99 66 99 AA FF FF
.:7AA8 99 66 99 66 99 AA FF FF
.:7AB0 DD 77 DD 77 DD FF AA AA
.:7AB8 DD 77 DD 77 DD FF AA AA
.:7AC0 99 66 99 66 99 AA FF FF
.:7AC8 DD 77 DD 77 DD FF AA AA
.:7AD0 DD 77 DD 77 DD FF AA AA
.:7AD8 DD 77 DD 77 DD FF AA AA
.:7AE0 DD 77 DD 77 DD 55 AA AA
.:7AE8 77 DD 77 DD 77 FF AA AA
.:7AF0 66 99 66 99 66 AA FF FF
.:7AF8 77 DD 77 DD 77 FF AA AA
.:7B00 77 DD 77 DD 77 FF AA AA
.:7B08 95 A5 E5 E5 E5 A5 F9 FE
.:7B10 FF FF FF FF FF FF FF FF
.:7B18 FF FF FF FF FF FF FF FF
.:7B20 FF FF FF FF FF FF FF FF
.:7B28 FF FF FF FF FF FF FF FF
.:7B30 D5 D5 D5 D5 BD AD AB AB
.:7B38 FF FF FF FF FF FF FF FF
.:7B40 A7 A9 AA AA AA AA AA AA
.:7B48 BB EE BB EE 7B 6E 7B 5E
.:7B50 99 66 99 66 99 66 99 66
.:7B58 2E 8B 2E 8B 2E 8B 2E 8B
.:7B60 E6 99 E6 99 E6 99 E6 99
.:7B68 EE BB EE BB EE BB EE BB
.:7B70 EE BB EE BB EE BB EE BB
.:7B78 E6 99 E6 99 E6 99 E6 99
.:7B80 AA AA AA AA AA AA AA AA
.:7B88 AA AA AA AA AA AA AA AA
.:7B90 AA AA AA AA AA AA AA AA
.:7B98 FF FF FF FF FF FF FF FF
.:7BA0 FF FF FF FF FF FF FF FF
.:7BA8 FF FF FF FF FF FF FF FF
.:7BB0 FF FF FF FF FF FF FF FF
.:7BB8 FF FF FF FF FF FF FF FF
```

```
.:7BC0 FF FF FF FF FF FF FF FF
.:7BC8 FF FF FF FF FF FF FF FF
.:7BD0 FF FF FF FF FF FF FF FF
.:7BD8 FF FF FF FF FF FF FF FF
.:7BE0 FF FF FF FF FF FF FF FF
.:7BE8 FF FF FF FF FF FF FF FF
.:7BF0 AA AA AA AA AA AA AA AA
.:7BF8 AA AA AA AA AA AA AA AA
.:7C00 FF FF FF FF FF FF FF FF
.:7C08 AA AA AA AA AA AA AA AA
.:7C10 AA AA AA AA AA AA AA AA
.:7C18 AA AA AA AA AA AA AA AA
.:7C20 AA AA AA AA AA AA AA AA
.:7C28 AA AA AA AA AA AA AA AA
.:7C30 AA AA AA AA AA AA AA AA
.:7C38 AA AA AA AA AA AA AA AA
.:7C40 AA AA AA AA AA AA AA AA
.:7C48 AB AB AB AB AA AA AA AA
.:7C50 55 55 55 55 D5 D5 D5 B5
.:7C58 55 55 55 55 55 55 55 55
.:7C60 FF FF FF FF FF FF FF FF
.:7C68 FF FF FF FF FF FF FF FF
.:7C70 FF FF FF FF FF FF FF FF
.:7C78 95 95 95 A5 E9 FE FF FF
.:7C80 AA AA AA AA AA AA EA EA
.:7C88 F6 FD FE FD FF FF FF FF
.:7C90 99 66 99 66 99 66 D9 E6
.:7C98 26 89 26 89 26 89 26 89
.:7CA0 D9 66 D9 66 D9 66 D9 66
.:7CA8 BB EE BB EE BB EE BB EE
.:7CB0 DD 77 DD 77 DD 77 DD 77
.:7CB8 E6 99 E6 99 E6 99 E6 99
.:7CC0 55 55 55 55 55 55 55 55
.:7CC8 55 55 55 55 55 55 55 55
.:7CD0 55 55 55 55 55 55 55 55
.:7CD8 55 55 55 55 55 55 55 55
.:7CE0 55 55 55 55 55 55 55 55
.:7CE8 55 55 55 55 55 55 55 55
.:7CF0 55 55 55 55 55 55 55 55
.:7CF8 55 55 55 55 55 55 55 55
.:7D00 AA AA AA AA AA AA AA AA
.:7D08 55 55 55 55 55 55 55 55
.:7D10 55 55 55 55 55 55 55 55
.:7D18 AA AA AA AA AA AA AA AA
.:7D20 55 55 55 55 55 55 55 55
.:7D28 FF FF FF FF FF FF FF FF
.:7D30 55 55 55 55 55 55 55 55
.:7D38 55 55 55 55 55 55 55 55
.:7D40 55 55 55 55 55 55 55 55
.:7D48 55 55 55 55 55 55 55 55
.:7D50 55 55 55 55 55 55 55 55
.:7D58 55 55 55 55 55 55 55 55
.:7D60 55 55 55 55 55 55 55 55
.:7D68 55 55 55 55 55 55 55 55
.:7D70 55 55 55 55 55 55 55 55
.:7D78 55 55 55 55 55 55 55 55
.:7D80 55 55 55 55 55 55 55 55
.:7D88 FF FF FF FF FF FF FF FF
.:7D90 B5 AD AD AB AB AA AA AA
.:7D98 55 55 55 55 55 D5 D5 B5
.:7DA0 FF FF FF FF FF FF FF FF
.:7DA8 FF FF FF FF FF FF FF FF
.:7DB0 FF FF FF FF FF FF FF FF
.:7DB8 FF FF FF FF FF FF FF FF
.:7DC0 95 A5 E9 FA FF FF FF FF
.:7DC8 AA AA AA AA EA EA FA 7A
.:7DD0 B7 AC A7 AC AB A8 AA AA
.:7DD8 3B CE 3B CE 3B CE 3B CE
.:7DE0 D9 66 D9 66 D9 66 D9 66
.:7DE8 77 DD 77 DD 77 DD 77 DD
.:7DF0 BB EE BB EE BB EE BB EE
.:7DF8 B7 DD B7 DD B7 DD B7 DD
.:7E00 FF FF FF FF FF FF FF FF
.:7E08 FF FF FF FF FF FF FF FF
.:7E10 AA AA AA AA AA AA AA AA
.:7E18 FF FF FF FF FF FF FF FF
.:7E20 AA AA AA AA AA AA AA AA
.:7E28 FF FF FF FF FF FF FF FF
.:7E30 AA AA AA AA AA AA AA AA
.:7E38 AA AA AA AA AA AA AA AA
.:7E40 55 55 55 55 55 55 55 55
.:7E48 FF FF FF FF FF FF FF FF
.:7E50 AA AA AA AA AA AA AA AA
.:7E58 AA AA AA AA AA AA AA AA
.:7E60 FF FF FF FF FF FF FF FF
.:7E68 FF FF FF FF FF FF FF FF
.:7E70 FF FF FF FF FF FF FF FF
.:7E78 FF FF FF FF FF FF FF FF
.:7E80 FF FF FF FF FF FF FF FF
.:7E88 FF FF FF FF FF FF FF FF
.:7E90 FF FF FF FF FF FF FF FF
.:7E98 FF FF FF FF FF FF FF FF
.:7EA0 FF FF FF FF FF FF FF FF
.:7EA8 55 55 55 55 55 55 55 55
.:7EB0 AA AA AA AA AA AA AA AA
.:7EB8 FF FF FF FF FF FF FF FF
.:7EC0 FF FF FF FF FF FF FF FF
.:7EC8 FF FF FF FF FF FF FF FF
.:7ED0 AA AA AA AA AA AA AA AA
.:7ED8 E5 F9 F9 FE FF FF FF FF
.:7EE0 55 55 55 55 95 95 E5 E5
.:7EE8 55 55 55 55 55 55 55 55
.:7EF0 AA AA AA AA AA AA AA AA
.:7EF8 AA AA AA AA AA AA AA AA
.:7F00 AA AA AA AA AA AA AA AA
.:7F08 BD AD AD AB AB AA AA AA
.:7F10 FF FF FF FF FF BF BF AA
.:7F18 26 89 26 89 26 89 26 89
.:7F20 D9 66 D9 66 D9 66 D9 66
.:7F28 66 99 66 99 66 99 66 99
.:7F30 99 66 99 66 99 66 99 66
.:7F38 E6 99 E6 99 E6 99 A6 99
.:7F40 B0 EB DE B9 92 E7 9D BE
.:7F48 E7 E7 5B EE 95 BE 91 5D
.:7F50 91 AE 00 EE 91 EE 00 92
.:7F58 11 EE 99 EE 10 EE 11 BE
.:7F60 17 78 87 78 87 87 99 BE
.:7F68 7E EE DE EE 9D E0 DE EA
```

```
.:7F70 CE EE EC EC 00 EE 81 EE
.:7F78 10 BE 9A EE 00 EC CE BE
.:7F80 0E EE 00 E0 10 B0 A0 70
.:7F88 78 87 E7 E8 E8 80 70 B7
.:7F90 E7 EE 80 EB 00 E7 02 E9
.:7F98 EE EE EC EE CE 7E 9D BE
.:7FA0 00 B9 00 EE CE EE EC EC
.:7FA8 EE BE 08 B0 90 EE B0 80
.:7FB0 87 78 E8 E8 E8 70 70 B7
.:7FB8 89 BE 9E 9E 09 E9 B9 B9
.:7FC0 95 9E 1E B9 9A E9 18 EE
.:7FC8 99 19 19 E9 90 EE E0 E9
.:7FD0 9E 79 79 CF C0 BC 0C 9E
.:7FD8 B9 89 8E E0 79 09 09 B2
.:7FE0 09 09 90 E9 90 E9 E9 09
.:7FE8 09 09 90 09 90 E9 9E 0E
.:7FF0 09 09 09 E9 09 09 09 09
.:7FF8 90 09 90 0C 0C B0 C0 B9
.:8000 09 09 09 0B 09 90 09 09
.:8008 E5 D5 95 E5 15 E5 15 45
.:8010 95 BE 05 B5 05 E7 5E E0
.:8018 E0 25 95 E5 5E B5 95 5E
.:8020 A5 B5 85 F0 0C CB 0C 50
.:8028 05 B5 B5 B0 B5 B0 B5 5B
.:8030 05 E9 95 B9 B5 B5 95 EE
.:8038 EE B5 BE 75 05 B5 B5 15
.:8040 95 25 B5 B5 15 05 95 95
.:8048 75 50 E0 FC C0 CB C0 0B
.:8050 CB CB 0B C0 0B 0C 0B CB
.:8058 09 05 95 E9 E0 B9 B9 B9
.:8060 0E BE 95 05 0E B0 B5 B5
.:8068 05 05 B0 E0 95 E0 90 7E
.:8070 E5 5E E9 0C 0C FC 0B CB
.:8078 BC 0C 0C 0C 0C 0C 0C 0B
.:8080 05 D5 05 09 0E B9 B9 5E
.:8088 D5 DE ED ED E0 B0 5B B0
.:8090 0B B5 D5 D5 05 0E E0 E0
.:8098 50 DE 0E C0 EC C0 F0 F0
.:80A0 0C 0C 0C B0 0B B0 CB 0B
.:80A8 E5 5E 05 B5 95 B5 B9 E9
.:80B0 0E BE 0E E0 05 E5 B5 B5
.:80B8 5E A0 10 B0 0E 5E 95 E5
.:80C0 E0 5E 1E CF 0E CF 0F B0
.:80C8 B0 B0 0B C0 B0 CB CB 0B
.:80D0 ED DE 0E E9 B9 B9 0E 5E
.:80D8 5E 5E 50 ED ED E0 E0 E0
.:80E0 05 5E 50 5E 05 5E 0E EE
.:80E8 ED ED 5E CF CE CF 0F CF
.:80F0 0F EB 0C FC 0F EC 0B CB
.:80F8 9E EE 9E B9 B5 5B 95 E5
.:8100 05 B5 05 E5 0E 95 0E BE
.:8108 05 DE 95 E5 15 50 0E 0E
.:8110 9E BE 0E C0 0C 0C 0F CF
.:8118 C0 B0 F0 FC CF CE CB CB
.:8120 AE BE E0 5B B5 A5 05 85
.:8128 05 50 E0 EE 9E BE 1E E0
.:8130 05 E5 95 B5 15 A5 E5 E0
.:8138 0E BE BE F0 0C C0 0C C0
.:8140 C0 C0 B0 CF CF CE C0 CB
.:8148 5E 5E B9 5B 5B 5E 5E 5E
.:8150 5E 50 E5 5E 5E 5E E5 E0
.:8158 5E 5E 5E 5E 5E 5E E0 5E
.:8160 5E 5E 5E CF 0C CF 0F CF
.:8168 CF CF CF CF FC CE CB CB
.:8170 05 5E B9 B9 E9 EB 09 E9
.:8178 E9 E5 15 E5 05 E5 15 5E
.:8180 0E EE 0E BE 1E E5 05 9E
.:8188 9E 1E 9E C0 C0 C0 0F 0F
.:8190 0F FC 0F FC 0F C0 0C B0
.:8198 5E EB B9 B9 E9 EE 9E E9
.:81A0 05 B5 05 E5 05 E5 9E E0
.:81A8 9E EE 0E BE 9E EE 0E 5E
.:81B0 0E BE E5 FC EC C0 0F CF
.:81B8 0F 0F 0F CF 0F EC B0 C0
.:81C0 19 EE B9 9B B9 EE E9 50
.:81C8 15 15 15 E5 95 E5 50 ED
.:81D0 95 EE 1E EE 0E EE 05 EE
.:81D8 9E E5 E5 FE EC C0 C0 E0
.:81E0 0F 0F 0F CF CF EC CB B0
.:81E8 E9 E0 B9 B9 9B B9 B9 50
.:81F0 50 50 05 50 05 50 05 E9
.:81F8 E9 E9 E9 E9 E9 E9 E9 E0
.:8200 5E 5E 0E E0 0C C5 05 50
.:8208 F0 0F F0 F0 F0 E0 B0 0B
.:8210 B5 B5 B5 B5 95 59 B9 B9
.:8218 B9 B9 9B B9 9B 9B 5B B5
.:8220 B5 B5 B5 B5 B5 B5 B5 EB
.:8228 0E EE 1E EE E5 E5 E5 5E
.:8230 0F 0C 0C FC C0 C0 CB CB
.:8238 B5 B5 B5 B5 B5 B9 9B 9B
.:8240 9B B9 95 B9 B9 5B 5B B5
.:8248 59 95 95 95 B5 B5 95 59
.:8250 90 AE 0E EE 9E 5E 05 5E
.:8258 CF 0F 0C FC C0 E0 C0 C0
.:8260 B5 B5 95 B5 B5 B9 9B B9
.:8268 5B B9 B5 B9 95 B9 5B B5
.:8270 B5 95 95 95 95 95 B5 B5
.:8278 59 EE 9E EE 0E 50 E5 E5
.:8280 05 F0 F0 CF C0 CE C0 C0
.:8288 B5 B5 95 5B B9 B9 9B B9
.:8290 5B B9 B9 B9 B9 B9 B5 B5
.:8298 B9 B5 B5 B5 95 B5 B9 B5
.:82A0 B5 E9 0E EE 0E BE 5E EE
.:82A8 F5 5F 0F C0 C0 E0 C0 C0
.:82B0 05 B5 05 E0 0E B9 0E BE
.:82B8 8E BE 1E BE 0E E0 95 B5
.:82C0 9E B5 15 B5 15 75 15 E5
.:82C8 95 E5 E5 EE 9E 7E 9E 50
.:82D0 E5 F0 0F F0 0C EC 0C C0
.:82D8 5E 5E 50 5E 5E 5E 5E 5E
.:82E0 E5 5E 5E D5 5E E0 5E 5E
.:82E8 5E 5E 5E 5E 5D 5E 5E 5E
.:82F0 5E E0 E5 E5 E5 5E E5 5E
.:82F8 50 E5 F5 CF 0C CE CB CB
.:8300 0E E0 05 EE 05 9E 05 B5
.:8308 5E DE 95 E5 1E E0 1E 2E
.:8310 9E BE 0E BE 0E 5E 15 BE
.:8318 8E E0 95 E0 E0 E0 9E BE
```

```
.:8320 8E 5E E0 F0 0C C0 0B C0
.:8328 FE FE FE FE FE FE FE FE
.:8330 FE FE FE FE FE FE FE FE
.:8338 FE 9E FE FE FE FE FE FE
.:8340 8E FE FE FE FE DE 4E AE
.:8348 FE 0E 2E FE 6E 0E 0E FE
.:8350 0E DE FE FE 2E 0E FE 0E
.:8358 9E 6E FE 4E 8E FE 4E 0E
.:8360 AE 0E 4E AE AE FE 8E CE
.:8368 DE FE AE 8E 0E FE CE 0E
.:8370 EE DE 08 17 40 AE 1E FE
.:8378 FE 6E 0E AE FE 9E 0E FE
.:8380 F9 DC 6E FC AE 1E 0E FE
.:8388 8E DE FE 8E BE FC FE 0E
.:8390 49 FE FE FE AE F0 FE AE
.:8398 4E 2E F7 A7 00 EE 7E 8E
.:83A0 4E 4E 9E BE FE 80 0E EE
.:83A8 5E 0E AE 4E FE FE AE 4E
.:83B0 FE 6E 1E 1E FE 19 E9 F7
.:83B8 A7 CE 6E F0 BE A0 FB 9E
.:83C0 1E 1E F9 09 80 FE FE FE
.:83C8 DE AE EE B0 2E 20 40 AE
.:83D0 AE FE AE FE 4E 50 A0 19
.:83D8 4E BE 8E C0 4E 9E FE FE
.:83E0 B7 F7 C7 AF FE AC 2B 00
.:83E8 FB AB 3B F9 9B 0B 0E FE
.:83F0 2E FE FE 99 69 FE FE AE
.:83F8 AE F5 FE 4E EE B5 FE F5
.:8400 B5 CE 50 D0 BE 40 AE FE
.:8408 0E 4E 2E CC FE B0 6B 5B
.:8410 FB 60 10 F5 F0 F5 E0 90
.:8418 FE 55 0E 45 F9 89 DB F5
.:8420 B5 0E D5 FE 4E F0 F0 F0
.:8428 F0 B0 F0 A0 C0 1E F0 2E
.:8430 0E EE F5 E0 EE 10 FB DC
.:8438 00 F0 DC 9B 4C AB FC 00
.:8440 45 A9 BE F5 D5 05 F5 D5
.:8448 25 F5 5E 4E 55 D5 F0 20
.:8450 0B 4B F5 95 B0 F5 95 65
.:8458 F0 20 8E 3F FE A0 2F E0
.:8460 F0 6B FB 2B FB FB 4B 1C
.:8468 F9 19 0E 05 F5 45 95 E0
.:8470 4E 25 F5 15 05 05 F0 55
.:8478 25 F0 40 F0 FE 05 45 F5
.:8480 FE FE FE FF F0 FF FB FB
.:8488 5B FB FB FC FC FC A0 FC
.:8490 F9 F9 49 F9 FE F9 F5 F5
.:8498 85 F5 F5 F5 FE F0 F0 F0
.:84A0 F0 F5 95 F5 F5 F0 EE F0
.:84A8 FE FE FE F0 FC F0 FC FF
.:84B0 FF FC FC 0F FC F0 F0 FC
.:84B8 F5 F5 45 F5 F5 D5 45 F9
.:84C0 F0 F0 FE F5 F5 F5 F5 F5
.:84C8 5E F0 FE F0 FE F0 45 F0
.:84D0 FE FE 4E F0 F0 F0 FC F0
.:84D8 4B F0 FB F0 FC F0 FC F0
.:84E0 F5 F5 C5 F5 F9 F9 EE F9
.:84E8 FE FE FE FE F5 FE F5 F5
.:84F0 FE F5 FE EE FE FE F5 FE
.:84F8 FE FE 4E FF FE 6F 4C F0
.:8500 FF FF FB F0 F0 F0 F0 F0
.:8508 55 F5 F5 F9 F9 F9 E9 F9
.:8510 F9 FE 45 F5 F5 F5 F5 F5
.:8518 0E FE FE FE BE 6E F0 AE
.:8520 9E 3E FE FC 6E FF 4F 2F
.:8528 FF 0F 5F 00 00 F0 EB C0
.:8530 4D FD E5 69 49 09 A9 FD
.:8538 F9 9E 1D FD 0D BD FD E5
.:8540 4D DD FD BD 2D 9D F5 4E
.:8548 6E FE 9E 60 0E F0 4C 20
.:8550 F0 80 F0 40 80 00 40 40
.:8558 5E F9 05 F5 45 25 05 F5
.:8560 E5 8E 4E 6E 4E FE CE FD
.:8568 05 F5 45 85 F5 0E 1E FE
.:8570 FE 0E 1E DF FE BF 6C FC
.:8578 0C 80 AC F0 AC 0E FB 4C
.:8580 89 45 65 95 65 45 B5 45
.:8588 F9 FE FE FE BE FE F5 15
.:8590 45 85 25 F5 F5 B5 05 FE
.:8598 FE 0E FE 50 60 FF 0C C0
.:85A0 0C FC FC 00 2C F0 FC DB
.:85A8 15 F5 A5 A5 F5 85 85 4E
.:85B0 09 39 FE 4E BE FE FE A5
.:85B8 8E F5 E5 F5 85 05 4E 00
.:85C0 AE FE 9E 00 40 FF 8F ED
.:85C8 FC 9C 8C F0 F0 B0 00 FC
.:85D0 A5 D5 E5 F5 F5 D5 15 F9
.:85D8 19 89 F9 B9 09 49 E9 A5
.:85E0 05 45 B5 15 F5 D5 F5 FE
.:85E8 9E AE CE 5F 45 A0 6F FE
.:85F0 8C 9C 4C FC AC AC FC 0C
.:85F8 C9 F9 A9 A9 FB FB 25 95
.:8600 E5 15 65 F5 D5 F5 89 29
.:8608 F9 49 29 49 E9 09 09 49
.:8610 AE A0 FE CE D0 F0 F0 F0
.:8618 FC FF EF F0 FF FE F0 F0
.:8620 F9 F9 F9 F9 F9 F5 F5 F5
.:8628 F5 45 FB F5 F5 F9 F9 09
.:8630 FB FB FB FB A9 B9 FB AB
.:8638 FE AE FE 0E 0E E0 4E 20
.:8640 F0 6C 8F D0 FF 0C 6B FB
.:8648 09 F9 9B F9 49 E5 85 F5
.:8650 89 C5 F9 D5 DB F5 59 09
.:8658 59 FB AB FB 1B FB 89 A9
.:8660 4B 20 8E FE FE 0E 90 A0
.:8668 FF AC DC E0 BF A0 FB 9B
.:8670 A9 B9 FB 29 85 F5 85 A5
.:8678 A9 F5 D5 85 F5 F5 A9 89
.:8680 F5 89 09 89 5B A9 D5 F9
.:8688 79 AB FE FE AE AE 40 F5
.:8690 40 F0 0C BF 4F FC FB 9B
.:8698 DE EE 6E 05 F5 A5 05 D5
.:86A0 F5 B5 15 F5 05 B5 2E FE
.:86A8 05 AE FE DE 8E 4E FE 0E
.:86B0 9E A0 50 10 9E FE 4E 0E
.:86B8 E0 F5 35 CC 4F F0 1B FB
.:86C0 6D 0D EE FD DD A9 FD ED
.:86C8 9D CD FD 2E 2D 65 FD CD
```

```
.:86D0 3D FD CD 0D AE FD 8D AD
.:86D8 FD 85 D0 40 AE 2E AE 4E
.:86E0 5E 20 B0 F0 EF 20 90 F0
.:86E8 A5 D5 4E 25 FE F5 0E FE
.:86F0 4D F5 AE DE F5 D5 A5 F5
.:86F8 F5 15 A5 C5 F5 BD AE F5
.:8700 05 35 2E F5 F5 FE FE AE
.:8708 AE 40 E5 8C AF 4E 2C 1B
.:8710 01 00 FF FF 00 00 FF FF
.
```

Listing C-17: The SPRITE MAKER Program

```
1 REM SAVE"@0:MAKE SPRITE",8
5 P=0:BASE=23552:CO=1:CL=0:CK=0
10 REM CLIMB SCREEN MAKER
20 POKE 53269,3
25 IF PEEK(32760)=1THEN GOTO43
30 IF A=0 THEN A=1:LOAD "SLIB.O",8,1
40 IF A=1 THEN A=2:LOAD "CLSP2",8,1
43 INPUT"WHAT IS THE FILE NAME";N$
50 POKE 56578,PEEK(56578)OR3:REM BANK1
52 FOR X=0TO7:POKE 24576+X,0:POKE 24576+8+X,255:NEXT
60 POKE 56576,(PEEK(56576)AND 252)OR 2
70 POKE 53272,120
80 POKE 53287,7:POKE 53288,7
81 POKE 53275,0
85 SX=24:SY=50:OX=0:OY=0
90 POKE 53248,24:POKE 53249,50
91 BC=6:POKE 53280,BC:POKE53281,BC
92 POKE 53250,255:POKE 53251,50
95 IF PEEK(32768)=1 THEN GOTO 132
96 IF Q=2 THEN GOTO 121
100 SYS 32768+3
110 SYS 32768+6
121 POKE 32760,1
130 FOR X=1 TO 8:POKE 23552+X+1015,0:NEXTX
131 POKE 24568,1
132 POKE 53287,0
140 REM LOOP
150 J1=PEEK(56320):J1=255-J1
160 J2=PEEK(56321):J2=255-J2
170 F= J1 AND 16
180 IF F = 0 THEN GET A$
190 IF A$="{??}"THEN P=ABS(1-P)
200 IF A$="{??}"THEN P=P
201 IF A$="{??}"THEN CK=PEEK(53285):CK=(CK+1)AND15:POKE 53285,CK
202 IF A$="{??}"THEN CO=((CO+1)AND15):POKE 53288,CO
203 IF A$="{??}"THEN BC=((BC+1)AND15)
204 IF A$="{??}"THEN CL=PEEK(53286):CL=(CL+1)AND15:POKE 53286,CL
205 IF A$="{??}"THEN POKE53276,2
206 IF A$="{??}"THEN POKE53276,0
207 IF A$="S" THEN GOSUB2000
208 IF A$="L" THEN GOSUB3000
209 IF A$="C" THEN GOSUB4000
219 A$=""
220 IF(J1 AND 1)= 1 THEN OY=OY-1
230 IF(J1 AND 2)= 2 THEN OY=OY+1
240 IF(J1 AND 4)= 4 THEN OX=OX-1
250 IF(J1 AND 8)= 8 THEN OX=OX+1
270 IF OX<0 THEN OX=23
280 IF OX>23 THEN OX=0
290 IF OY<3 THEN OY=24
300 IF OY>23 THEN OY=3
```

```
301 SX=24+(8*OX):SY=50+(8*OY)
302 IF SX<256 THEN POKE 53264,0
303 IF SX>255 THEN SX=SX-256:POKE 53264,3
310 POKE 53248,SX:POKE 53249,SY
320 IF C=0THEN C=1:POKE 53287,1
330 IF C=1THEN C=0:POKE 53287.0
340 POKE BASE,P
350 POKE 53280,BC:POKE 53281,BC
359 IF F=16 THEN POKE BASE+(OY*40)+OX,P:POKE 55296+(OY*40)+OX,1
360 SYS32768
490 GOTO 140
500 REM THIS IS 500--SAVE THE SCREEN + COLOR
2000 REM WRITE FILE SUBROUTINE
2010 OPEN 8,8,8,"@0:"+N$+".SP,S,W"
2020 FOR X=0 TO 1000
2030 PRINT#8,CHR$(PEEK(23552+X));
2040 NEXT X
2050 CLOSE 8
2900 RETURN
3000 REM READ FILE SUBROUTINE
3010 OPEN 8,8,8,N$+".SP,S,R"
3020 FOR X=0 TO 1000
3030 GET#8,DB$
3035 IF DB$="" THEN DB$=CHR$(0)
3036 POKE 23552+X,ASC(DB$)
3040 NEXT X
3050 CLOSE 8
3900 RETURN
4000 REM CLEAR SCREEN
4010 SYS32768+3
4020 SYS32768+6
4030 RETURN
5000 REM BANK0
5060 POKE 56578,PEEK(56578)OR3
5070 POKE 56576,(PEEK(56576)AND 252)OR3
5080 POKE 53272,20
5200 RETURN
5500 REM BANK1
5560 POKE 56578,PEEK(56578)OR3
5570 POKE 56576,(PEEK(56576)AND 252)OR2
5580 POKE 53272,120
5600 RETURN
6000 REM WORK ON NEW SPRITE
6010 GOSUB 5000
6020 INPUT "WHICH SPRITE DO YOU WANT TO WORK ON";SP
6030 GOSUB5500
6040 SYS 32768+3
6900 RETURN
```

Listing C-18: The SLIB.O File

```
B*
   PC  SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:8000 4C 2A 80 4C 92 80 4C AE
.:8008 80 40 01 02 04 08 10 20
.:8010 40 80 FE FD FB F7 EF DF
.:8018 BF 7F FE FD FB F7 EF DF
.:8020 BF 7F 80 40 20 10 08 04
.:8028 02 01 48 98 48 8A 48 A9
.:8030 00 AA A9 78 85 FB A9 5C
.:8038 85 FC A9 00 85 FD 85 FE
```

```
.:8040 A8 8D FE 7F 8D FF 7F 48
.:8048 B1 FB F0 05 68 1D 22 80
.:8050 48 E8 C8 E0 08 D0 F1 A6
.:8058 FE 68 9D 00 40 E6 FE A2
.:8060 00 8A 48 EE FE 7F AD FE
.:8068 7F C9 03 D0 DB EE FF 7F
.:8070 A9 00 8D FE 7F A5 FB 18
.:8078 69 28 85 FB A9 00 65 FC
.:8080 85 FC A0 00 AD FF 7F C9
.:8088 15 D0 BD 68 68 AA 68 A8
.:8090 68 60 A9 00 85 FD A9 D8
.:8098 85 FE A0 00 A2 00 A9 01
.:80A0 91 FD C8 D0 FB E6 FE A5
.:80A8 FE C9 DC D0 F1 60 A9 00
.:80B0 85 FD A9 5C 85 FE A0 00
.:80B8 A2 00 A9 00 91 FD C8 D0
.:80C0 FB E6 FE E8 E0 04 D0 F2
.:80C8 60 FF FF FF FF FF 02 FF
.
.
```

Listing C-19: The CLSP2 File

```
B*
   PC  SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.

.:4000 FF 00 00 FF 00 00 FF 00
.:4008 00 FF 00 00 FF 00 00 FF
.:4010 00 00 FF 00 00 FF 00 00
.:4018 00 00 00 00 00 00 00 00
.:4020 00 00 00 00 00 00 00 00
.:4028 00 00 00 00 00 00 00 00
.:4030 00 00 00 00 00 00 00 00
.:4038 00 00 00 00 00 00 00 00
.:4040 FF 00 00 FF 00 00 FF 00
.:4048 00 FF 00 00 FF 00 00 FF
.:4050 00 00 FF 00 00 FF 00 00
.:4058 00 00 00 00 00 00 00 00
.:4060 00 00 00 00 00 00 00 00
.:4068 00 00 00 00 00 00 00 00
.:4070 00 00 00 00 00 00 00 00
.:4078 00 00 00 00 00 00 00 00
.:4080 FF FF 00 00 FF BF 00 00
.:4088 FF FF 00 00 FF FF 00 00
.:4090 FF FF 00 00 FF FF 00 00
.:4098 FF FF 00 00 FF FF 00 00
.:40A0 FF FF 00 00 FF BF 00 00
.:40A8 FF FF 00 00 FF FF 00 00
.:40B0 FF FF 00 00 FF FF 00 00
.:40B8 FF FF 00 00 FF FF 00 00
.:40C0 FF FF 00 00 FF BF 00 00
.:40C8 FF FF 00 00 FF FF 00 00
.:40D0 FF FF 00 00 FF FF 00 00
.:40D8 FF FF 00 00 FF FF 00 00
.:40E0 FF FF 00 00 FF FF 00 00
.:40E8 FF FF 00 00 FF FF 00 00
.:40F0 FF FF 00 00 FF FF 00 00
.:40F8 FF FF 00 00 FF FF 00 00
.:4100 00 00 FF FF 00 00 FF FF
.
```

Listing C-20: The SCREEN-MAKE Program

```
1 REM SAVE"@0:MAKE SCREEN",8
5 P=0:BASE=23552:CO=1
10 REM CLIMB SCREEN MAKER
20 POKE 53269,3
25 IF PEEK(32760)=1THEN GOTO41
30 IF A=0 THEN A=1:LOAD "CLBACK1",8,1
40 IF A=1 THEN A=2:LOAD "CLSP1",8,1
41 IF A=2 THEN A=3:LOAD "SLIB.O",8,1
43 INPUT"WHAT IS THE FILE NAME";N$
50 POKE 56578,PEEK(56578)OR3:REM BANK1
60 POKE 56576,(PEEK(56576)AND 252)OR 2
70 POKE 53272,120
80 POKE 53287,7:POKE 53288,7
81 POKE 53275,3
85 SX=24:SY=50:OX=0:OY=0
90 POKE 53248,24:POKE 53249,50
91 BC=6:POKE 53280,BC:POKE53281,BC
95 REM IF PEEK(32760)=1 THEN GOTO 132
96 REM IF Q=2 THEN GOTO 121
100 SYS 32768+3:SYS 32768+6
```

```
121 POKE 32760,1
130 FOR X=1 TO 8:POKE 23552+X+1015,0:NEXTX
132 POKE 53287,0
140 REM LOOP
150 J1=PEEK(56320):J1=255-J1
160 J2=PEEK(56321):J2=255-J2
170 F= J1 AND 16
180 IF F = 0 THEN GET A$
190 IF A$="{??}"THEN P=P+1
200 IF A$="{??}"THEN P=P-1
201 IF A$="{??}"THEN GOTO 500
202 IF A$="{??}"THEN CO=((CO+1)AND 15)
203 IF A$="{??}"THEN BC=((BC+1)AND15)
204 IF A$="S"THEN GOSUB 1000
205 IF P<0 THEN P=0
206 IF A$="L"THEN GOSUB 2000
207 IF A$="C" THEN GOSUB3000
210 A$=""
220 IF(J1 AND 1)= 1 THEN OY=OY-1
230 IF(J1 AND 2)= 2 THEN OY=OY+1
240 IF(J1 AND 4)= 4 THEN OX=OX-1
250 IF(J1 AND 8)= 8 THEN OX=OX+1
260 IF F=16 THEN POKE BASE+(OY*40)+OX,P:POKE 55296+(OY*40)+OX,CO
270 IF OX<0 THEN OX=39
280 IF OX>39 THEN OX=0
290 IF OY<0 THEN OY=24
300 IF OY>24 THEN OY=0
301 SX=24+(8*OX):SY=50+(8*OY)
302 IF SX<256 THEN POKE 53264,0
303 IF SX>255 THEN SX=SX-256:POKE 53264,3
310 POKE 53248,SX:POKE 53249,SY
320 IF C=0THEN C=1:POKE 53287,1
330 IF C=1THEN C=0:POKE 53287,0
340 POKE BASE,P:POKE 55296,CO
350 POKE 53280,BC:POKE 53281,BC
490 GOTO 140
500 REM THIS IS 500--SAVE THE SCREEN + COLOR
510 REM SYS28672
1000 REM SAVE FILE
1010 OPEN 8,8,8,"@0:"+N$+".BC,S,W"
1020 FOR X=0 TO 1000
1030 PRINT#8,CHR$(PEEK(23552+X));CHR$(PEEK(55296+X));
1040 NEXT X
1050 CLOSE 8
1900 RETURN
2000 REM LOAD FILE
2010 OPEN 8,8,8,N$+".BC,S,R"
2020 FOR X=0 TO 1000
2030 GET#8,DB$,DC$
2035 IF DB$=""THENDB$=CHR$(0)
2036 IF DC$=""THENDC$=CHR$(0)
2040 POKE 23552+X,ASC(DB$):POKE55296+X,ASC(DC$)
2050 NEXTX
2060 CLOSE8
2900 RETURN
3000 REM CLEAR SCREEN
3010 SYS32768+3
3020 SYS32768+6
3030 RETURN
```

Listing C-21: The CLBACK1 File

```
B*
    PC   SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:6000 00 00 00 00 00 00 00 00
.:6008 FF 03 06 0C 18 30 60 FF
.:6010 FF C0 60 30 18 0C 03 FF
.:6018 C0 C0 FF C0 C0 C0 FF C0
.:6020 03 03 FF 03 03 03 FF 03
.:6028 FF 03 06 0C 18 30 C0 FF
.:6030 FF C0 60 30 18 0C 03 FF
.:6038 00 00 00 00 00 00 00 00
.:6040 00 00 00 00 00 00 00 00
.:6048 00 00 FF 03 06 0C 18 30
.:6050 00 00 FF C0 60 30 18 0C
.:6058 C0 FF 00 00 00 00 00 00
.:6060 03 FF 00 00 00 00 00 00
.:6068 00 00 00 00 FF 03 06 0C
.:6070 00 00 00 00 FF C0 60 30
.:6078 18 30 C0 FF 00 00 00 00
.:6080 18 0C 03 FF 00 00 00 00
.:6088 00 00 00 00 00 00 FF 03
.:6090 00 00 00 00 00 00 FF C0
.:6098 06 0C 18 30 C0 FF 00 00
.:60A0 60 30 18 0C 03 FF 00 00
.:60A8 00 00 00 00 00 00 C0 C0
.:60B0 00 00 00 00 C0 C0 C0 C0
.:60B8 00 00 C0 C0 C0 C0 C0 C0
.:60C0 C0 C0 C0 C0 C0 C0 C0 C0
.:60C8 00 00 00 00 00 00 00 00
.:60D0 00 00 18 66 C3 C3 66 18
.:60D8 10 20 40 80 58 24 08 10
.:60E0 20 40 30 C0 02 01 02 0C
.:60E8 10 20 30 08 18 18 66 C3
.:60F0 07 01 00 00 00 00 00 00
.:60F8 3F 0F 00 00 00 00 00 00
.:6100 FF 1F 03 00 00 00 00 00
.:6108 FF FF FF 00 00 00 00 00
.:6110 FF FF F8 E0 C0 00 00 00
.:6118 FF 81 00 00 00 00 00 00
.:6120 FF FF 1F 07 03 00 00 00
.:6128 00 00 00 00 00 00 00 00
.:6130 00 BA 22 FF 00 FF 00 62
.:6138 00 B8 22 FF 00 FF BA FF
.:6140 00 00 FF FF 00 00 FF FF
.
.
```

Listing C-22: The CLSP1 File

```
B*
    PC   SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:4000 FF 00 00 FF 00 00 FF 00
.:4008 00 FF 00 00 FF 00 00 FF
.:4010 00 00 FF 00 00 FF 00 00
.:4018 00 00 00 00 00 00 00 00
.:4020 00 00 00 00 00 00 00 00
.:4028 00 00 00 00 00 00 00 00
.:4030 00 00 00 00 00 00 00 00
.:4038 00 00 00 00 00 00 00 00
.:4040 00 FF 00 FF 00 FF 22 FF
.:4048 05 FF 00 FF 3F FF 00 FF
.:4050 FF FF 00 00 FF FF 00 00
.:4058 FF FF 00 00 FF FF 00 00
.:4060 FF FF 00 00 FF FF 00 00
.:4068 FF FF 00 00 FF FF 00 00
.:4070 FF FF 00 00 FF FF 00 00
.:4078 FF FF 00 00 FF FF 00 00
.:4080 FF FF 00 00 FF BF 00 00
.:4088 FF FF 00 00 FF FF 00 00
.:4090 FF FF 00 00 FF FF 00 00
.:4098 FF FF 00 00 FF FF 00 00
.:40A0 FF FF 00 00 FF BF 00 00
.:40A8 FF FF 00 00 FF FF 00 00
.:40B0 FF FF 00 00 FF FF 00 00
.:40B8 FF FF 00 00 FF FF 00 00
.:40C0 FF FF 00 00 FF BF 00 00
.:40C8 FF FF 00 00 FF FF 00 00
.:40D0 FF FF 00 00 FF FF 00 00
.:40D8 FF FF 00 00 FF FF 00 00
.:40E0 FF FF 00 00 FF FF 00 00
.:40E8 FF FF 00 00 FF FF 00 00
.:40F0 FF FF 00 00 FF FF 00 00
.:40F8 FF FF 00 00 FF FF 00 00
.:4100 00 00 FF FF 00 00 FF FF
.
.
```

Listing C-23: The PHOENIX V1.4N Program

```
.:1000 01 01 01 01 01 01 01 01
.:1008 01 01 01 01 01 01 01 01
.:1010 01 01 01 01 01 01 01 01
.:1018 01 01 01 01 01 01 01 01
```

```
.:1020 01 01 01 01 01 01 01 01
.:1028 00 00 00 00 00 00 00 00
.:1030 00 03 00 00 00 00 00 01
.:1038 00 00 00 00 03 03 00 03
.:1040 00 08 00 00 00 00 00 00
.:1048 00 00 00 03 00 00 00 07
.:1050 00 00 00 00 00 00 01 00
.:1058 00 00 01 01 01 01 01 01
.:1060 01 01 01 0D 01 01 01 01
.:1068 01 01 01 01 00 00 00 00
.:1070 00 00 00 00 00 00 00 00
.:1078 00 00 00 00 03 00 07 01
.:1080 01 01 01 01 03 03 03 03
.:1088 03 03 03 03 03 03 03 03
.:1090 03 03 01 01 01 01 01 00
.:1098 00 00 00 00 00 00 00 00
.:10A0 01 01 01 01 01 01 01 01
.:10A8 03 03 03 03 03 00 01 00
.:10B0 00 00 00 00 00 00 00 00
.:10B8 00 03 03 03 03 03 01 01
.:10C0 01 01 00 00 00 00 00 00
.:10C8 00 00 01 01 01 01 03 03
.:10D0 03 00 00 00 00 00 00 00
.:10D8 00 00 00 00 00 0D 00 00
.:10E0 00 00 00 00 00 03 03 03
.:10E8 01 01 01 01 00 00 03 08
.:10F0 09 09 01 01 03 03 01 01
.:10F8 00 00 00 00 00 00 00 00
.:1100 00 00 07 00 00 00 00 03
.:1108 00 01 00 00 00 00 00 00
.:1110 03 0D 01 01 01 01 00 01
.:1118 08 09 09 03 0F 03 0C 03
.:1120 0F 02 0F 03 0F 00 0F 00
.:1128 0F 02 0F 00 0B 00 07 0F
.:1130 0F 03 0F 00 0F 00 0F 00
.:1138 0B 00 03 0D 01 09 09 09
.:1140 08 09 08 09 09 09 09 0F
.:1148 0F 00 0F 03 0F 00 0F 03
.:1150 0F 00 0F 00 0F 00 01 00
.:1158 0F 00 0B 0F 0F 00 0B 09
.:1160 09 09 09 09 09 08 07 08
.:1168 08 08 08 08 08 08 08 0F
.:1170 0F 00 00 00 0F 00 0F 00
.:1178 0F 00 04 00 03 00 07 00
.:1180 00 05 09 0D 08 0F 09 09
.:1188 07 07 07 09 09 08 08 08
.:1190 08 08 08 08 08 08 09 09
.:1198 08 0F 00 0B 00 0F 0B 0F
.:11A0 08 0F 00 0F 00 0F 00 0F
.:11A8 00 0F 00 0F 0F 09 09 08
.:11B0 09 07 07 09 09 08 08 08
.:11B8 08 08 08 08 08 09 08 08
.:11C0 09 09 00 0F 00 0F 00 0F
.:11C8 08 0F 00 0F 08 0F 09 09
.:11D0 09 09 09 09 09 09 08 08
.:11D8 08 09 08 08 08 09 09 07
.:11E0 09 08 08 09 09 09 09 08
.:11E8 09 08 09 0F 00 0F 03 0C
.:11F0 00 0F 0F 0F 00 09 09 09
.:11F8 07 08 09 09 09 08 08 09
.:1200 08 08 08 08 09 07 09 08
.:1208 09 08 08 08 08 08 08 08
.:1210 08 08 09 0F 0F 00 0F 01
.:1218 0F 08 0F 04 09 09 09 08
.:1220 08 08 08 08 09 08 09 07
.:1228 07 09 09 09 09 09 08 09
.:1230 09 08 08 08 08 08 08 08
.:1238 09 08 08 09 03 03 0F 03
.:1240 0F 08 09 09 09 09 08 09
.:1248 09 08 08 08 08 07 09 09
.:1250 09 09 09 09 09 09 09 08
.:1258 09 09 08 08 08 08 08 08
.:1260 08 08 08 09 09 0F 0F 00
.:1268 0F 09 09 09 09 08 08 09
.:1270 07 09 08 08 08 08 08 08
.:1278 09 09 07 07 07 09 09 09
.:1280 09 09 08 08 08 08 08 08
.:1288 08 08 09 08 09 0F 02 09
.:1290 09 09 09 08 07 09 09 09
.:1298 08 08 09 09 08 08 08 09
.:12A0 09 09 07 07 07 09 08 09
.:12A8 09 09 08 08 08 09 08 08
.:12B0 08 08 08 08 08 09 00 09
.:12B8 09 08 08 07 08 08 08 08
.:12C0 09 09 09 09 08 08 09 09
.:12C8 07 07 09 09 09 09 09 09
.:12D0 09 09 08 08 08 08 08 08
.:12D8 08 08 08 08 08 09 09 09
.:12E0 09 08 08 08 08 07 08 08
.:12E8 09 09 09 09 09 09 09 09
.:12F0 09 09 07 07 09 09 09 09
.:12F8 09 09 09 08 08 08 09 08
.:1300 08 08 08 08 08 08 08 08
.:1308 08 08 08 08 08 08 09 09
.:1310 08 09 09 09 07 09 09 09
.:1318 09 09 09 08 08 08 09 09
.:1320 09 0F 0C 0F 0C 08 08 08
.:1328 08 08 09 09 09 08 08 08
.:1330 08 09 08 08 09 09 08 08
.:1338 09 09 09 08 08 08 08 08
.:1340 09 09 09 07 07 08 08 08
.:1348 09 0F 0C 0F 09 0A 09 09
.:1350 09 09 09 0F 09 09 09 08
.:1358 08 09 09 09 09 08 09 09
.:1360 09 09 09 09 09 09 09 07
.:1368 09 09 09 09 09 09 09 09
.:1370 09 0F 09 0C 0C 0F 09 0C
.:1378 0C 0F 0F 0C 0C 09 09 09
.:1380 09 09 0F 09 09 09 09 09
.:1388 09 07 0C 07 07 09 09 09
.:1390 09 09 09 0F 0F 0C 0C 09
.:1398 09 0C 09 0C 0C 09 09 09
.:13A0 09 09 0F 0C 05 05 0F 0C
.:13A8 0F 0C 0F 07 09 09 07 07
.:13B0 09 09 0F 09 0F 09 0F 09
.:13B8 0C 09 05 05 05 09 09 09
.:13C0 09 0F 0F 0C 0C 09 09 09
.:13C8 09 09 09 0C 0C 0C 0C 0F
```

```
.:13D0 0C 0C 0C 0C 0C 0C 0C 0C
.:13D8 0F 0C 0F 0F 0C 0C 0C 0C
.:13E0 0F 0C 0C 0C 09 0F 0C 09
.:13E8 00 0F 04 0F 02 0F 00 01
.:13F0 00 0F 00 0F 00 0F 00 0F
.:13F8 0F 0D 0F 0D 0D 0D 0D 0D
.:1400 E8 E8 82 E2 E2 E2 E2 E2
.:1408 E2 E2 E2 E2 E2 E2 E2 E2
.:1410 E8 E2 E2 E2 E2 E8 E8 E8
.:1418 E2 E2 E2 E2 E2 E2 E2 E2
.:1420 E2 E2 E2 E2 E8 E8 E8 E8
.:1428 E8 E8 82 E2 E2 E2 E2 E2
.:1430 E2 E2 E2 E2 E2 E2 E2 E2
.:1438 E2 E2 E1 E2 E8 E8 E8 E8
.:1440 E2 E2 E2 E2 E2 E2 E2 E2
.:1448 E2 E2 E2 E2 E8 E8 E8 E8
.:1450 E8 E8 E2 E8 E2 E2 E2 E2
.:1458 E2 E2 E2 E2 E2 E2 E8 E2
.:1460 E1 E2 E8 E2 E8 E8 E8 E8
.:1468 E2 E2 E2 E2 E2 E2 E2 E2
.:1470 E2 E2 E2 E2 E8 E8 E8 E8
.:1478 E8 E8 E8 E8 E8 E8 E8 E8
.:1480 E8 E8 E8 E8 E8 E8 E8 E8
.:1488 E8 E8 E8 E8 E8 E8 E8 E8
.:1490 E8 E8 E8 E8 E8 E8 E8 E8
.:1498 E8 E8 E8 E8 E8 E8 E8 E8
.:14A0 E8 E8 E8 E8 E8 E8 E8 E8
.:14A8 E8 E8 E8 E8 E8 E8 E8 E8
.:14B0 E8 E8 E8 E8 E8 E8 E8 E8
.:14B8 E8 E8 E8 E8 E8 E8 E8 E8
.:14C0 E8 E8 E8 E8 E8 E8 E8 E8
.:14C8 E8 E8 E8 E8 E8 E8 E8 E8
.:14D0 E8 E8 E8 E8 E8 E8 E8 E8
.:14D8 E8 E8 E8 E8 E8 E8 E8 E8
.:14E0 E8 E8 E8 E8 E8 E8 E8 E8
.:14E8 E8 E8 E8 E8 E8 E8 E8 E8
.:14F0 E8 E8 E8 E8 E8 E8 E8 E8
.:14F8 E8 E8 E8 E8 E8 E8 E8 E8
.:1500 E8 E8 E8 E8 E8 E8 E8 E8
.:1508 E8 E8 E8 E8 E8 E8 E8 E8
.:1510 E8 E8 E8 E8 E8 E8 E8 E8
.:1518 E8 E8 E8 E8 E8 E8 E8 E8
.:1520 E8 E8 E8 E8 E8 E8 E8 E3
.:1528 13 13 13 13 13 E3 E8 E8
.:1530 E8 E8 E8 E8 E8 E8 E8 E8
.:1538 E8 E8 E8 E8 E8 E8 E8 E8
.:1540 E8 E8 E8 E8 E8 E8 E8 E8
.:1548 E8 E8 E8 E8 E8 E8 E8 13
.:1550 13 E1 E1 E1 E1 13 E8 E8
.:1558 E8 E8 E8 E8 E8 E8 E8 E8
.:1560 E8 E8 E8 E8 E8 E8 E8 E8
.:1568 E8 E8 E8 E8 E8 E8 E8 E8
.:1570 E8 E8 E8 E8 E8 E8 E8 13
.:1578 E1 13 13 13 13 13 E8 E8
.:1580 E8 E8 E8 E8 E8 E8 E8 E8
.:1588 E8 E8 E8 E8 E8 E8 E8 E8
.:1590 E8 E8 E8 E8 E8 E8 E8 E8
.:1598 E8 E8 E8 E8 E8 E8 E8 13
.:15A0 13 13 13 13 13 13 E8 E8
.:15A8 E8 E8 E8 E8 E8 E8 E8 E8
.:15B0 E8 E8 E8 E8 E8 E8 E8 E8
.:15B8 E8 E8 E8 E8 E8 E8 E8 E8
.:15C0 E8 E8 E8 E8 E8 E8 E3 E3
.:15C8 13 13 13 13 13 13 E3 E8
.:15D0 E8 E8 E8 E8 E8 E8 E8 E8
.:15D8 E8 E8 E8 E8 E8 E8 E8 E8
.:15E0 E8 E8 E8 E8 E8 E8 E8 E8
.:15E8 E8 E8 E8 E8 E8 E8 E3 13
.:15F0 13 E1 E1 E1 E1 13 E3 E8
.:15F8 E8 E8 E8 E8 E8 E8 E8 E8
.:1600 E8 E8 E8 E8 E8 E8 E8 E8
.:1608 E8 E8 E8 E8 E8 E8 E8 E8
.:1610 E8 E8 E8 E8 E8 E8 E3 13
.:1618 E1 E1 E1 E1 13 13 E3 E8
.:1620 E8 E8 E8 E8 E8 E8 E8 E8
.:1628 E8 E8 E8 E8 E8 E8 E8 E8
.:1630 E8 E8 E8 E8 E8 E8 E8 E8
.:1638 E8 E8 E8 E8 E8 E8 E3 13
.:1640 13 E1 E1 E1 13 13 E3 E3
.:1648 E8 E8 E8 E8 E8 E8 E8 E8
.:1650 E8 E8 E8 E8 E8 E8 E8 E8
.:1658 E8 E8 E8 E8 E8 E8 E8 E8
.:1660 E8 E8 E8 E8 E8 E8 E3 E8
.:1668 13 E1 E1 E1 E1 13 E8 E3
.:1670 E8 E8 E8 E8 E8 E8 E8 E8
.:1678 E8 E8 E8 E8 E8 E8 E8 E8
.:1680 E8 E8 E8 E8 E8 E8 E8 E8
.:1688 E8 E8 E8 E8 E8 E8 E8 E3
.:1690 E3 13 E1 13 13 E3 E3 E8
.:1698 E8 E8 E8 E8 E8 E8 E8 E8
.:16A0 E8 E8 E8 E8 E8 E8 E8 E8
.:16A8 E8 E8 E8 E8 E8 E8 E8 E8
.:16B0 E8 E8 E8 E8 E8 E8 E8 E3
.:16B8 E3 13 13 13 E3 E3 E8 E8
.:16C0 E8 E8 E8 E8 E8 E8 E8 E8
.:16C8 E8 E8 E8 E8 E8 E8 E8 E8
.:16D0 E8 E8 E1 E8 E1 E1 E1 E1
.:16D8 E1 E1 E1 E8 E8 E8 E8 13
.:16E0 13 E1 E1 E1 13 13 E1 E1
.:16E8 E8 E8 E8 E8 E1 E1 E1 E1
.:16F0 E1 E1 E1 E1 E1 E8 E8 E8
.:16F8 E8 E8 E1 E8 E1 E1 E1 E1
.:1700 E1 E1 E1 E8 E8 E8 E8 E1
.:1708 E1 E1 E1 E1 E1 E1 E1 E1
.:1710 E8 E8 E8 E8 E1 E1 E1 E1
.:1718 E1 E1 E1 E1 E1 E8 E8 E8
.:1720 E8 E8 E8 51 E1 E1 A4 E1
.:1728 E1 E1 E8 E8 E8 E8 E8 E1
.:1730 E1 E1 E5 45 E8 E8 E8 E1
.:1738 E8 E8 E8 E8 E1 E1 E1 E5
.:1740 45 E8 E8 E8 E1 E8 E8 E8
.:1748 E8 E8 E5 51 54 E8 E4 14
.:1750 E8 E1 E8 E8 E8 E8 E1 E1
.:1758 E1 E1 E4 E4 E5 E8 E8 E1
.:1760 E8 E8 E8 E8 E1 E1 E1 E4
.:1768 E4 E5 E8 E8 E1 E8 E8 E8
.:1770 E8 E8 E8 51 E1 E1 E4 E1
.:1778 E1 E1 E8 E8 E8 E8 E8 E1
```

```
.:1780 E1 E1 E5 E5 E1 E1 E1 E1
.:1788 E8 E8 E8 E8 E1 E1 E1 E5
.:1790 E5 E1 E1 E1 E1 E8 E8 E8
.:1798 E8 E1 E1 E1 E1 E1 E1 E1
.:17A0 E1 E1 E1 E1 E1 E1 E1 E1
.:17A8 E1 E1 E1 E1 E1 E1 E1 E1
.:17B0 E1 E8 E1 E1 E1 E1 E1 E1
.:17B8 E1 E1 E8 E1 E1 E1 E1 E1
.:17C0 E8 E1 E1 E1 E1 E1 E1 E1
.:17C8 E1 E1 E8 E1 E1 E1 E1 E1
.:17D0 E1 E1 E8 E1 E1 E1 E1 E1
.:17D8 E1 E1 E1 E1 E1 E1 E1 E1
.:17E0 E1 E1 E1 E1 E1 E1 E1 E1
.:17E8 AA AA AA AA AA AA AA AA
.:17F0 AA AA AA AA AA AA AA AA
.:17F8 AA AA AA AA AA AA AA AA
.:1800 E2 F2 F3 F3 E3 F3 F3 03
.:1808 43 F3 F3 F3 F3 F3 43 51
.:1810 F2 F1 43 F3 F3 F2 42 F2
.:1818 F3 F3 F3 F3 F3 F3 F3 F3
.:1820 F3 F3 F3 F3 E3 F3 F3 D3
.:1828 42 F2 F3 F3 F3 F3 63 53
.:1830 F3 F3 43 F3 F3 F3 43 F3
.:1838 F1 F1 F2 F3 F2 F2 F2 F2
.:1840 F3 F3 F3 F3 E3 F3 F3 03
.:1848 43 F3 F3 F3 F2 F3 53 53
.:1850 F3 F2 43 F2 F3 F3 43 F3
.:1858 F3 F3 F3 F3 F3 F3 F2 F1
.:1860 A2 F1 F2 F3 E2 02 F2 A3
.:1868 43 F3 03 F3 83 13 03 53
.:1870 A3 A3 43 83 02 D3 43 D3
.:1878 03 F2 03 D3 93 F3 F3 83
.:1880 F3 E3 F3 03 F3 F3 03 03
.:1888 F3 F3 73 33 F3 83 83 A3
.:1890 F3 D3 03 F3 F3 A3 63 F3
.:1898 F3 F3 93 23 F3 B3 B3 F3
.:18A0 03 B2 A2 F2 D2 A2 F2 F2
.:18A8 92 82 F2 E2 F2 A2 F2 B2
.:18B0 82 F2 42 02 A2 02 42 B2
.:18B8 32 F2 F3 F3 43 F3 D3 D3
.:18C0 F3 F3 03 B3 E3 83 A3 E3
.:18C8 43 93 03 F3 F3 83 A3 83
.:18D0 F3 B3 43 F3 F3 D3 23 F2
.:18D8 43 B3 03 F3 A3 03 F3 F3
.:18E0 63 23 F3 B3 03 F3 F3 03
.:18E8 63 C3 F3 A3 A3 43 F3 A3
.:18F0 B3 E3 F3 F3 E3 43 A3 B3
.:18F8 F3 63 03 F3 A3 A3 03 F3
.:1900 B3 B3 F3 03 A3 F3 D3 83
.:1908 D3 F3 F3 93 A3 F3 13 93
.:1910 03 F3 03 B3 F3 33 C3 43
.:1918 F3 13 43 43 03 03 F3 D3
.:1920 A3 F3 83 F3 C3 13 43 20
.:1928 60 F0 C0 30 80 F0 03 23
.:1930 F3 83 A3 F3 83 D3 03 F3
.:1938 23 03 E3 F3 A3 03 63 F3
.:1940 23 53 F3 03 A3 F3 D3 A3
.:1948 43 F3 F3 43 E3 43 D3 F0
.:1950 60 C3 B3 F3 F3 40 03 43
.:1958 D3 93 02 42 82 A2 F2 42
.:1960 92 F2 A2 F2 B2 52 03 A3
.:1968 F3 D3 B3 F3 F3 F3 C3 E3
.:1970 F3 63 F3 03 93 03 F3 B0
.:1978 93 F0 00 40 60 F0 43 23
.:1980 13 F3 22 B3 F2 22 02 D2
.:1988 52 42 82 F2 42 22 03 F3
.:1990 A2 12 42 42 82 F2 F2 F2
.:1998 F2 82 F2 F2 F2 E2 F2 F2
.:19A0 F2 F2 F2 F2 F2 F2 F2 F2
.:19A8 F2 F2 82 F2 F2 F2 F2 F2
.:19B0 F2 42 F2 F2 02 42 F2 F2
.:19B8 F2 F2 F2 F2 F2 F2 F2 52
.:19C0 F3 F2 F9 F2 F2 E2 F2 F2
.:19C8 F2 49 F9 F2 F2 F2 F2 F2
.:19D0 F2 F2 F2 F3 F3 F3 F3 F3
.:19D8 F3 D3 F3 F3 F3 E3 F3 F2
.:19E0 F2 F2 F2 F2 F2 F2 F2 F2
.:19E8 F3 F2 D3 F2 F2 F2 F2 F2
.:19F0 F2 49 F9 F2 A2 02 42 E2
.:19F8 82 F2 C3 23 D3 53 F3 03
.:1A00 F3 F3 83 A3 23 53 A3 A2
.:1A08 F2 B2 02 62 F2 32 52 42
.:1A10 D3 A2 F2 42 A2 22 F2 F2
.:1A18 E2 49 E9 92 02 F2 02 82
.:1A20 92 F2 42 92 F2 A3 82 82
.:1A28 F3 C3 03 F3 C3 93 43 F2
.:1A30 B2 43 F3 53 03 93 F3 03
.:1A38 E3 F3 F3 63 23 43 F0 60
.:1A40 F9 B9 E3 43 F0 C0 B0 F0
.:1A48 E3 B3 A3 F3 C3 03 A3 F3
.:1A50 D3 63 F3 F3 13 03 F3 82
.:1A58 F2 F3 D3 43 43 C3 13 03
.:1A60 43 53 53 03 F3 89 A9 F9
.:1A68 F9 49 23 43 43 C0 F0 A0
.:1A70 53 63 F3 53 A3 F3 F3 93
.:1A78 F3 F3 A3 E3 03 F3 43 E2
.:1A80 F2 23 43 E3 F3 43 03 F3
.:1A88 93 F3 43 03 43 A3 43 10
.:1A90 90 F0 F3 00 80 80 F0 E3
.:1A98 E3 43 03 A3 F3 A3 63 23
.:1AA0 F3 A3 23 F3 03 A3 F3 02
.:1AA8 A2 02 F2 F2 A2 12 F2 02
.:1AB0 52 22 F2 F2 22 F2 E2 E2
.:1AB8 42 A2 02 A2 42 C2 02 F2
.:1AC0 22 62 F2 82 F2 12 C2 42
.:1AC8 E2 A2 F2 F2 22 42 F2 02
.:1AD0 83 F3 93 A3 F3 A3 13 43
.:1AD8 F3 43 A3 E3 F3 D3 A3 40
.:1AE0 F0 83 F3 F3 F0 80 43 A3
.:1AE8 F3 D3 43 03 E3 F3 33 F3
.:1AF0 F3 F3 F3 F3 F3 F3 F3 F3
.:1AF8 F3 F3 F3 F3 F3 F3 F3 F3
.:1B00 F3 F3 F3 F3 F3 F3 F3 F3
.:1B08 F3 F3 13 F3 F3 F3 F3 F3
.:1B10 F3 43 F3 F3 03 43 F3 F3
.:1B18 F3 F3 F3 F3 F3 F3 F3 53
.:1B20 F3 F3 F3 FA FA 43 F5 FA
.:1B28 F3 43 F3 F3 F3 F3 F3 F3
```

```
.:1B30 F3 F3 FA FA F1 F1 F1 F3
.:1B38 F3 F3 F3 F3 F3 E3 F3 FA
.:1B40 FA F1 F1 F1 F3 F3 F3 F3
.:1B48 F3 F3 91 F4 F1 F5 F5 F5
.:1B50 F1 43 F3 F3 83 43 13 03
.:1B58 F3 83 25 F5 61 21 B1 53
.:1B60 73 83 F3 03 23 03 F3 F5
.:1B68 05 41 11 41 F3 F3 03 03
.:1B70 E3 F3 23 54 E4 13 15 F4
.:1B78 B3 B3 03 F3 63 03 F3 83
.:1B80 03 E3 F4 A4 A3 F3 F3 B3
.:1B88 23 F3 23 A3 A3 53 B3 04
.:1B90 F4 03 13 D3 F3 83 C3 43
.:1B98 A3 B3 F3 13 83 F3 F3 F3
.:1BA0 A3 B3 E3 D3 43 F3 F3 F3
.:1BA8 B3 F3 E3 D3 F3 03 83 43
.:1BB0 F3 93 A3 F3 A3 A3 43 F3
.:1BB8 83 93 81 53 93 03 F3 43
.:1BC0 B3 13 F3 A3 83 43 93 83
.:1BC8 F3 33 63 03 F3 03 A3 F3
.:1BD0 E3 23 13 F3 A3 F3 93 F3
.:1BD8 03 E3 F3 D3 F3 43 F3 A3
.:1BE0 83 F3 F3 A3 43 63 F3 D3
.:1BE8 04 1A 60 A9 01 85 1D 56
.:1BF0 19 23 13 A5 1A 38 E9 05
.:1BF8 9D 11 1D 1D 11 1D F3 FF
.:1C00 03 A0 00 03 F0 00 03 CA
.:1C08 00 03 1A 01 43 F3 00 83
.:1C10 1B 01 83 7C 00 43 7C 00
.:1C18 83 F4 00 03 2D 00 03 7D
.:1C20 00 43 CD 00 83 F5 00 83
.:1C28 1D 01 43 1D 01 83 CE 00
.:1C30 03 57 00 43 A7 00 83 CF
.:1C38 00 83 1F 01 83 A8 00 83
.:1C40 D0 00 83 31 00 83 A9 00
.:1C48 03 21 01 43 82 00 83 AA
.:1C50 00 03 33 00 43 83 00 83
.:1C58 AB 00 83 23 01 43 23 01
.:1C60 83 84 00 83 AC 00 83 85
.:1C68 00 83 86 00 03 37 00 03
.:1C70 5F 00 83 87 00 83 88 00
.:1C78 03 D8 00 83 89 00 83 8A
.:1C80 00 83 8B 00 03 DB 00 83
.:1C88 3C 00 43 3C 00 83 8C 00
.:1C90 83 3D 00 03 3D 00 83 8D
.:1C98 00 03 2D 01 83 8E 00 83
.:1CA0 3F 00 43 3F 00 03 67 00
.:1CA8 83 8F 00 83 07 01 83 90
.:1CB0 00 03 41 00 83 91 00 83
.:1CB8 B9 00 83 31 01 43 31 01
.:1CC0 43 92 00 83 BA 00 03 43
.:1CC8 00 43 93 00 83 BB 00 03
.:1CD0 33 01 83 BC 00 83 BD 00
.:1CD8 83 E5 00 43 BE 00 83 E6
.:1CE0 00 83 E7 00 03 37 01 43
.:1CE8 E8 00 83 10 01 03 11 01
.:1CF0 03 C2 00 43 12 01 83 3A
.:1CF8 01 83 4B 00 43 4B 00 03
.:1D00 3B 01 83 EE 00 43 EE 00
.:1D08 0D 52 00 4D F2 00 4D 1A
.:1D10 01 4D CB 00 0D F3 00 4D
.:1D18 CC 00 4D F4 00 4D A5 00
.:1D20 0D CD 00 4D A6 00 4D CE
.:1D28 00 0D A7 00 4D CF 00 4D
.:1D30 80 00 4D A8 00 4D 81 00
.:1D38 4D A9 00 0D 82 00 4D AA
.:1D40 00 0D 83 00 4D 5C 00 4D
.:1D48 84 00 4D 5D 00 4D 85 00
.:1D50 4D 5E 00 4D 86 00 4D 5F
.:1D58 00 4D 87 00 4D 60 00 4D
.:1D60 88 00 4D 61 00 4D 62 00
.:1D68 8D 63 00 4D 64 00 4D 65
.:1D70 00 4D 8D 00 8D DD 00 4D
.:1D78 DD 00 4D 66 00 4D 8E 00
.:1D80 4D 67 00 0D 8F 00 4D 68
.:1D88 00 4D 90 00 4D 69 00 4D
.:1D90 91 00 0D 92 00 0D 93 00
.:1D98 0D BB 00 4D 94 00 4D BC
.:1DA0 00 4D 95 00 4D BD 00 0D
.:1DA8 BE 00 4D E6 00 4D BF 00
.:1DB0 4D E7 00 4D C0 00 0D E8
.:1DB8 00 4D E9 00 8D 11 01 4D
.:1DC0 EA 00 0D 12 01 0D 4B 00
.:1DC8 4D 13 01 8D 3B 01 4D 3C
.:1DD0 01 0D 4E 00 0D 9E 00 81
.:1DD8 A0 00 41 18 01 81 A1 00
.:1DE0 41 F1 00 01 19 01 81 A2
.:1DE8 00 81 CA 00 81 F2 00 01
.:1DF0 2B 00 01 7B 00 81 A3 00
.:1DF8 81 CB 00 81 F3 00 01 1B
.:1E00 01 81 A4 00 81 CC 00 81
.:1E08 A5 00 81 CD 00 81 56 00
.:1E10 41 56 00 01 56 00 81 A6
.:1E18 00 81 F6 00 81 7F 00 81
.:1E20 A7 00 81 F7 00 81 80 00
.:1E28 81 81 00 81 5A 00 81 82
.:1E30 00 81 5B 00 81 83 00 81
.:1E38 5C 00 81 5D 00 81 5E 00
.:1E40 81 AE 00 41 AE 00 81 37
.:1E48 00 41 37 00 81 5F 00 81
.:1E50 60 00 01 88 00 81 61 00
.:1E58 01 3A 00 81 62 00 01 63
.:1E60 00 01 2B 01 81 64 00 01
.:1E68 B4 00 41 3D 00 81 65 00
.:1E70 81 66 00 81 67 00 81 68
.:1E78 00 81 69 00 81 09 01 41
.:1E80 09 01 81 6A 00 81 92 00
.:1E88 81 6B 00 81 93 00 81 94
.:1E90 00 81 95 00 01 BD 00 01
.:1E98 0D 01 81 96 00 81 BE 00
.:1EA0 81 BF 00 81 C0 00 81 E8
.:1EA8 00 01 49 00 81 C1 00 81
.:1EB0 E9 00 01 72 00 81 EA 00
.:1EB8 81 12 01 01 3A 01 81 EB
.:1EC0 00 81 13 01 81 14 01 81
.:1EC8 3C 01 81 15 01 01 3D 01
.:1ED0 01 C6 00 01 3E 01 81 17
.:1ED8 01 41 17 01 FF FF FF 7F
```

```
.:1EE0 B2 FF 00 ED 30 ED 00 FF
.:1EE8 00 FF 00 FF F1 FF 85 B1
.:1EF0 20 ED 00 FF 00 FF 80 DF
.:1EF8 00 FF FF FF FD FF 7D FF
.:1F00 FF 82 FF 00 FF 42 E7 E7
.:1F08 FF F7 FF 00 EF 00 FF F7
.:1F10 FF 10 FF 00 FF FF FF 00
.:1F18 FF F3 FF 30 FF 00 CE F3
.:1F20 FF 40 FF FF FF C3 FF 00
.:1F28 FF 00 FF 10 FF FF FF F3
.:1F30 76 FF FF 00 FF 00 FF 00
.:1F38 FF 00 FF 00 FF 01 FF B2
.:1F40 FF F7 FF 00 FF 00 FF FF
.:1F48 FF E7 FF FF FF 56 CF F7
.:1F50 FF 10 FF 00 FF 00 7F 50
.:1F58 FF 00 FF 10 FF 00 FF F3
.:1F60 FF 00 FF F3 FF F3 FF 00
.:1F68 FF 00 FF 00 FF 00 FF CF
.:1F70 FF F3 FF 00 FF 00 7F 20
.:1F78 FF 20 DF 30 FF 01 FF 00
.:1F80 00 FD 00 FF 70 FF 18 FD
.:1F88 00 7D 00 FF D1 FF 41 FD
.:1F90 11 FF 00 FF 00 FF 00 FF
.:1F98 00 CF 00 FF F9 FF FF FD
.:1FA0 00 FF 00 CC 04 FD 00 FF
.:1FA8 FB FF 00 FF E1 ED 85 FD
.:1FB0 FF FD 00 FF 00 FF 00 FF
.:1FB8 00 FF 00 FF 7D FF 7D 6D
.:1FC0 00 FD 30 FF F3 FF 04 ED
.:1FC8 00 7D 20 00 E5 EF FB ED
.:1FD0 11 FF FB FF 00 FF 80 FF
.:1FD8 30 FF 00 FF F9 FF F7 7F
.:1FE0 B2 FF 00 ED 30 ED 00 FF
.:1FE8 20 FF 00 FF F1 FF 85 B1
.:1FF0 20 ED 00 FF 00 FF 80 FF
.:1FF8 00 FF FF FF FD FF 7D FF
.:2000 00 00 00 00 00 00 00 00
.:2008 00 00 00 00 00 00 00 00
.:2010 00 00 00 00 00 00 02 0A
.:2018 00 00 00 00 00 2A AA AA
.:2020 00 00 00 00 00 AA AA AA
.:2028 00 00 00 00 00 AA AA A8
.:2030 00 00 00 00 00 80 02 02
.:2038 00 00 00 00 00 80 80 80
.:2040 00 00 00 00 00 00 00 00
.:2048 00 00 00 00 00 00 28 20
.:2050 00 00 00 00 00 00 2A 2A
.:2058 00 00 00 00 00 00 AA AA
.:2060 00 00 00 00 00 00 A0 A0
.:2068 00 00 00 00 00 00 0A 2A
.:2070 00 00 00 00 00 00 A0 A8
.:2078 00 00 00 00 00 00 00 0C
.:2080 00 00 00 00 00 00 00 00
.:2088 00 00 00 00 00 00 30 00
.:2090 00 00 00 00 00 00 00 00
.:2098 00 00 00 00 00 00 0A 2A
.:20A0 00 00 00 00 00 00 A0 A8
.:20A8 00 00 00 00 00 00 03 03
.:20B0 00 00 00 00 00 00 00 00
.:20B8 00 00 00 00 00 00 C0 C0
.:20C0 00 00 00 00 00 00 00 00
.:20C8 00 00 00 00 00 00 00 00
.:20D0 00 00 00 00 00 00 00 00
.:20D8 00 00 00 00 00 00 00 00
.:20E0 00 00 00 00 00 00 00 00
.:20E8 00 00 00 00 00 00 00 00
.:20F0 00 00 00 00 00 00 00 00
.:20F8 00 00 00 00 00 00 00 00
.:2100 00 00 00 00 00 00 00 00
.:2108 00 00 00 00 00 00 00 00
.:2110 00 00 00 00 00 00 00 00
.:2118 00 00 00 00 00 00 00 00
.:2120 00 00 00 00 00 00 00 00
.:2128 00 00 00 00 00 00 00 00
.:2130 00 00 00 00 00 00 00 00
.:2138 00 00 00 00 00 00 00 00
.:2140 00 00 00 00 00 00 00 00
.:2148 00 00 00 03 03 0F 3F 3C
.:2150 2A A8 A8 A0 80 00 00 00
.:2158 80 00 00 00 00 00 00 00
.:2160 00 00 00 00 00 00 00 00
.:2168 00 00 00 00 00 00 02 02
.:2170 0A 08 28 28 A0 A0 80 80
.:2178 80 80 80 80 80 80 80 A0
.:2180 00 00 02 02 02 0A 0A 28
.:2188 A0 80 80 00 00 00 00 00
.:2190 00 00 00 00 02 02 0A 08
.:2198 20 20 A0 80 80 00 00 00
.:21A0 00 00 00 00 02 02 0A 08
.:21A8 28 A0 A0 A0 A8 2A 0A 02
.:21B0 0A 02 02 0A 28 A0 00 00
.:21B8 00 00 02 02 08 08 08 08
.:21C0 2A 80 00 08 0C 2E BB 2E
.:21C8 00 80 20 20 08 08 88 08
.:21D0 00 00 80 00 03 03 03 0F
.:21D8 28 20 A0 A0 A8 8A 00 00
.:21E0 0F 03 03 0F 3C F0 00 00
.:21E8 03 03 03 03 03 03 03 03
.:21F0 03 03 0F 0C 3C F0 F0 C0
.:21F8 C0 00 00 00 00 00 00 00
.:2200 00 00 00 00 00 00 00 00
.:2208 00 00 00 00 00 00 00 00
.:2210 00 00 00 A8 82 82 82 A8
.:2218 00 00 00 0A 08 08 08 0A
.:2220 00 00 00 80 20 20 20 80
.:2228 00 00 00 AA 80 80 80 A8
.:2230 00 00 00 02 08 08 08 08
.:2238 00 00 00 82 02 02 02 02
.:2240 00 00 00 A8 00 00 00 A0
.:2248 00 00 00 20 20 28 28 22
.:2250 00 00 00 20 20 20 20 20
.:2258 00 00 00 A8 20 20 20 20
.:2260 00 00 00 0F 30 30 30 30
.:2268 00 00 00 00 00 00 00 00
.:2270 00 00 00 00 00 00 00 00
.:2278 00 00 00 00 00 00 00 00
.:2280 00 00 00 00 00 00 00 00
.:2288 3C F0 F0 F0 FF FF FF F0
```

```
.:2290 00 00 00 00 AA AA AA 00
.:2298 00 00 00 00 FF FF FF 00
.:22A0 00 00 00 00 A0 80 00 00
.:22A8 0A 0A 0A 28 28 A0 A0 00
.:22B0 00 00 00 00 00 00 00 00
.:22B8 20 20 28 08 0A 0A 0A 00
.:22C0 28 A0 A0 A0 80 80 00 00
.:22C8 00 00 00 00 00 00 00 00
.:22D0 08 28 20 20 A0 80 80 00
.:22D8 00 00 00 00 00 02 02 00
.:22E0 08 28 20 A0 80 80 00 00
.:22E8 00 02 02 0A 28 A0 A0 00
.:22F0 C0 00 00 00 00 00 00 00
.:22F8 08 C2 02 00 00 03 00 00
.:2300 08 0C 00 C0 3F 00 00 00
.:2308 08 20 20 80 00 02 C2 00
.:2310 0C 3C 30 F0 C0 C0 00 00
.:2318 00 00 00 00 00 00 00 00
.:2320 00 00 00 00 00 00 00 00
.:2328 03 0F 0C 3C 30 F0 C0 00
.:2330 00 00 00 00 00 00 00 00
.:2338 00 00 00 00 00 00 00 00
.:2340 00 00 00 00 00 00 00 00
.:2348 00 00 00 00 00 00 00 00
.:2350 80 80 80 80 80 80 00 00
.:2358 08 08 08 08 08 08 00 00
.:2360 80 20 20 20 20 20 00 00
.:2368 80 80 80 80 80 AA 00 00
.:2370 02 00 00 00 00 0A 00 00
.:2378 02 82 82 82 82 02 00 00
.:2380 00 00 00 00 00 A8 00 00
.:2388 22 22 20 20 20 20 00 00
.:2390 20 20 A0 A0 20 20 00 00
.:2398 20 20 20 20 20 20 00 00
.:23A0 0C 03 03 03 03 3C 00 00
.:23A8 00 00 00 00 00 00 00 00
.:23B0 00 00 00 00 00 00 00 00
.:23B8 00 00 00 00 00 00 00 00
.:23C0 00 00 00 00 00 00 00 00
.:23C8 F0 F0 F0 F0 3C 3C 3C 3C
.:23D0 00 00 00 00 00 00 00 00
.:23D8 00 00 00 00 00 00 00 00
.:23E0 00 00 00 00 00 00 00 00
.:23E8 00 00 00 00 00 00 00 00
.:23F0 00 00 00 00 00 00 00 00
.:23F8 00 00 00 00 00 00 00 00
.:2400 00 00 00 00 00 00 00 00
.:2408 00 00 00 00 00 00 00 00
.:2410 00 00 00 00 00 00 00 00
.:2418 00 00 00 00 00 00 00 00
.:2420 00 00 00 00 00 00 00 00
.:2428 00 00 00 00 00 00 00 00
.:2430 00 00 00 00 00 00 00 00
.:2438 00 00 00 00 00 00 00 00
.:2440 00 00 00 00 00 00 00 00
.:2448 00 00 00 00 00 00 00 00
.:2450 00 00 00 00 00 00 00 00
.:2458 00 00 00 00 00 00 00 00
.:2460 00 00 00 00 00 00 00 00
.:2468 00 00 00 00 00 00 00 00
.:2470 00 00 00 00 00 00 00 00
.:2478 00 00 00 00 00 00 00 00
.:2480 00 00 00 00 00 00 00 00
.:2488 00 00 00 00 00 00 00 00
.:2490 00 00 00 00 00 00 00 00
.:2498 00 00 00 00 00 00 00 00
.:24A0 00 00 00 00 00 00 00 00
.:24A8 00 00 00 00 00 00 00 00
.:24B0 00 00 00 00 00 00 00 00
.:24B8 00 00 00 00 00 00 00 00
.:24C0 00 00 00 00 00 00 00 00
.:24C8 00 00 00 00 00 00 00 00
.:24D0 00 00 00 00 00 00 00 00
.:24D8 00 00 00 00 00 00 00 00
.:24E0 00 00 00 00 00 00 00 00
.:24E8 00 00 00 00 00 00 00 00
.:24F0 00 00 00 00 00 00 00 00
.:24F8 00 00 00 00 00 00 00 00
.:2500 00 00 00 00 00 00 00 00
.:2508 3C 0F 0F 0F 03 00 00 00
.:2510 00 00 00 C0 F0 FF 3F 0F
.:2518 00 00 00 00 00 FF FF FF
.:2520 00 00 00 00 00 FF FF FF
.:2528 00 00 00 00 00 FF FF FF
.:2530 00 00 00 00 00 F0 C0 00
.:2538 03 03 0C 0C 30 30 C0 C0
.:2540 00 00 C3 C3 CC CC 30 30
.:2548 C0 C0 03 0C 0C 0C 0C 03
.:2550 30 CC 0C 00 3C 0C 30 C0
.:2558 03 00 03 03 0C 0C 30 FC
.:2560 F0 C0 00 00 03 03 0C 0C
.:2568 30 30 CC CC 0C 0C 03 03
.:2570 0C 0C 30 30 C0 C0 03 03
.:2578 0F 0C 30 3F C0 C0 00 FC
.:2580 F0 00 00 00 03 03 0C 0F
.:2588 3F 30 C0 FC 00 00 00 F0
.:2590 C0 00 03 03 0C 0C 30 30
.:2598 FC C3 C3 3C 30 0C 30 C0
.:25A0 03 00 03 03 0C 0C 30 FC
.:25A8 F0 C0 00 00 03 03 0C 0C
.:25B0 30 30 CC CC 0C 0C 03 03
.:25B8 0C 0C 30 30 C0 C0 00 00
.:25C0 03 0C 30 C0 C3 C0 C3 3C
.:25C8 00 C0 C0 00 C0 C0 00 00
.:25D0 00 00 00 00 00 00 00 00
.:25D8 00 00 00 00 00 00 00 00
.:25E0 00 00 00 00 00 00 00 00
.:25E8 00 00 00 00 00 00 00 00
.:25F0 00 00 00 00 00 00 00 00
.:25F8 00 00 00 00 00 00 00 00
.:2600 00 00 00 00 00 00 00 00
.:2608 00 00 00 00 00 00 00 00
.:2610 00 00 00 00 00 00 00 00
.:2618 00 00 00 00 00 00 00 00
.:2620 00 00 00 00 00 00 00 00
.:2628 00 00 00 00 00 00 00 00
.:2630 00 00 00 00 00 00 00 00
.:2638 00 00 00 00 00 00 00 00
```

```
.:2640 00 00 00 00 00 00 00 00
.:2648 00 00 00 00 00 00 00 00
.:2650 00 00 00 00 00 00 00 00
.:2658 00 00 00 00 00 00 00 00
.:2660 00 00 00 00 00 00 00 00
.:2668 00 00 00 00 00 00 00 00
.:2670 00 00 00 00 00 00 00 00
.:2678 00 00 00 00 00 00 00 00
.:2680 00 00 00 00 00 00 00 00
.:2688 00 00 00 00 00 00 00 00
.:2690 00 00 00 00 00 00 00 00
.:2698 00 00 00 00 00 00 00 00
.:26A0 00 00 00 00 00 00 00 00
.:26A8 00 00 00 00 00 00 00 00
.:26B0 00 00 00 00 00 00 00 00
.:26B8 00 00 00 00 00 00 00 00
.:26C0 00 00 00 00 00 00 00 00
.:26C8 00 00 00 00 00 00 00 00
.:26D0 00 00 00 00 00 00 00 00
.:26D8 00 00 00 00 00 00 00 00
.:26E0 00 00 00 00 00 00 00 00
.:26E8 00 00 00 00 00 00 00 00
.:26F0 00 00 00 00 00 00 00 00
.:26F8 00 00 00 00 00 00 00 00
.:2700 00 00 00 00 00 00 00 00
.:2708 00 00 00 00 00 00 00 00
.:2710 00 00 00 00 00 00 00 00
.:2718 00 00 00 00 00 00 00 00
.:2720 00 00 00 00 00 00 00 00
.:2728 00 00 00 00 00 00 00 00
.:2730 00 00 00 00 00 00 00 00
.:2738 00 00 00 00 00 00 00 00
.:2740 00 00 00 00 00 00 00 00
.:2748 00 00 00 00 00 00 00 00
.:2750 00 00 00 00 00 00 00 00
.:2758 00 00 00 00 00 00 00 00
.:2760 00 00 00 00 00 00 00 00
.:2768 00 00 00 00 00 00 00 00
.:2770 00 00 00 00 00 00 00 00
.:2778 00 00 00 00 00 00 00 00
.:2780 00 00 00 00 00 00 00 00
.:2788 00 00 00 00 00 00 00 00
.:2790 00 00 00 00 00 00 00 00
.:2798 00 00 00 00 00 00 00 00
.:27A0 00 00 00 00 00 00 00 00
.:27A8 00 00 00 00 00 00 00 00
.:27B0 00 00 00 00 00 00 00 00
.:27B8 00 00 00 00 00 00 00 00
.:27C0 00 00 00 00 00 00 00 00
.:27C8 00 00 00 00 00 00 00 00
.:27D0 00 00 00 00 00 00 00 00
.:27D8 00 00 00 00 00 00 00 00
.:27E0 00 00 00 00 00 00 00 00
.:27E8 00 00 00 00 00 00 00 00
.:27F0 00 00 00 00 00 00 00 00
.:27F8 00 00 00 00 00 00 00 00
.:2800 00 00 00 00 00 00 00 00
.:2808 00 00 00 00 00 00 00 00
.:2810 00 00 00 00 00 00 00 00
.:2818 00 00 00 00 00 00 00 00
.:2820 00 00 00 00 00 00 00 00
.:2828 00 00 00 00 00 00 00 00
.:2830 00 00 00 00 00 00 00 00
.:2838 00 00 00 00 00 00 00 00
.:2840 00 00 00 00 00 00 00 00
.:2848 00 00 00 00 00 00 00 00
.:2850 00 00 00 00 00 00 00 00
.:2858 00 00 00 00 00 00 00 00
.:2860 00 00 00 00 00 00 00 00
.:2868 00 00 00 00 00 00 00 00
.:2870 00 00 00 00 00 00 00 00
.:2878 00 00 00 00 00 00 00 00
.:2880 00 00 00 00 00 00 00 00
.:2888 00 00 00 00 00 00 00 00
.:2890 00 00 00 00 00 00 00 00
.:2898 00 00 00 00 00 00 00 00
.:28A0 00 00 00 00 00 00 00 00
.:28A8 00 00 00 00 00 00 00 00
.:28B0 00 00 00 00 00 00 00 00
.:28B8 00 00 00 00 00 00 00 00
.:28C0 00 00 00 00 00 00 00 00
.:28C8 00 00 00 00 00 00 00 00
.:28D0 00 00 00 00 00 00 00 00
.:28D8 00 00 00 00 00 00 00 00
.:28E0 00 00 00 00 00 00 00 00
.:28E8 00 00 00 00 00 00 00 00
.:28F0 00 00 00 00 00 00 00 00
.:28F8 00 00 00 00 00 00 00 00
.:2900 00 00 00 00 00 00 00 00
.:2908 00 00 00 00 00 00 00 00
.:2910 00 00 00 00 00 00 00 00
.:2918 00 00 00 00 00 00 00 00
.:2920 00 00 00 00 00 00 00 00
.:2928 00 00 00 00 00 00 00 00
.:2930 00 00 00 00 00 00 00 00
.:2938 00 00 00 00 00 00 00 00
.:2940 00 00 00 00 00 00 00 00
.:2948 00 00 00 00 00 00 00 00
.:2950 00 00 00 00 00 00 00 00
.:2958 00 00 00 00 00 00 00 00
.:2960 00 00 00 00 00 00 00 00
.:2968 00 00 00 00 00 00 00 00
.:2970 00 00 00 00 00 00 00 00
.:2978 00 00 00 00 00 00 00 00
.:2980 00 00 00 00 00 00 00 00
.:2988 00 00 00 00 00 00 00 00
.:2990 00 00 00 00 00 00 00 00
.:2998 00 00 00 00 00 00 00 00
.:29A0 00 00 00 00 00 00 00 00
.:29A8 00 00 00 00 00 00 00 00
.:29B0 00 00 00 00 00 00 00 00
.:29B8 00 00 00 00 00 00 00 00
.:29C0 00 00 00 00 00 00 00 00
.:29C8 00 00 00 00 00 00 00 00
.:29D0 00 00 00 00 00 00 00 00
.:29D8 00 00 00 00 00 00 00 00
.:29E0 00 00 00 00 00 00 00 00
.:29E8 00 00 00 00 00 00 00 00
```

```
.:29F0 00 00 00 00 00 00 00 00
.:29F8 00 00 00 00 00 00 00 00
.:2A00 00 00 00 00 00 00 00 00
.:2A08 00 00 00 00 00 00 00 00
.:2A10 00 00 00 00 00 00 00 00
.:2A18 00 00 00 00 00 00 00 00
.:2A20 00 00 00 00 00 00 00 00
.:2A28 00 00 00 00 00 00 00 00
.:2A30 00 00 00 00 00 00 00 00
.:2A38 00 00 00 00 00 00 00 00
.:2A40 00 00 00 00 00 00 00 00
.:2A48 00 00 00 00 00 00 00 00
.:2A50 00 00 00 00 00 00 00 00
.:2A58 00 00 00 00 00 00 00 00
.:2A60 00 00 00 00 00 00 00 00
.:2A68 00 00 00 00 00 00 00 00
.:2A70 00 00 00 00 00 00 00 00
.:2A78 00 00 00 00 00 00 00 00
.:2A80 00 00 00 00 00 00 00 00
.:2A88 00 00 00 00 00 00 00 00
.:2A90 00 00 00 00 00 00 00 00
.:2A98 00 00 00 00 00 00 00 00
.:2AA0 00 00 00 00 00 00 00 00
.:2AA8 00 00 00 00 00 00 00 00
.:2AB0 00 00 00 00 00 00 00 00
.:2AB8 00 00 00 00 00 00 00 00
.:2AC0 00 00 00 00 00 00 00 00
.:2AC8 00 00 00 00 00 00 00 00
.:2AD0 00 00 00 00 00 00 00 00
.:2AD8 00 00 00 00 00 00 00 00
.:2AE0 00 00 00 00 00 00 00 00
.:2AE8 00 00 00 00 00 00 00 00
.:2AF0 00 00 00 00 00 00 00 00
.:2AF8 00 00 00 00 00 00 00 00
.:2B00 00 00 00 00 00 00 00 00
.:2B08 00 00 00 00 00 00 00 00
.:2B10 00 00 00 00 00 00 00 00
.:2B18 00 00 00 00 00 00 00 00
.:2B20 00 00 00 00 00 00 00 00
.:2B28 00 00 00 00 00 00 00 00
.:2B30 00 00 00 00 00 00 00 00
.:2B38 00 00 00 00 00 00 00 00
.:2B40 00 00 00 00 00 00 00 00
.:2B48 00 00 00 00 00 00 00 00
.:2B50 00 00 00 00 00 00 00 00
.:2B58 00 00 00 00 00 00 00 00
.:2B60 00 00 00 00 00 00 00 00
.:2B68 00 00 00 00 00 00 00 00
.:2B70 00 00 00 00 00 00 00 00
.:2B78 00 00 00 00 00 00 00 00
.:2B80 00 00 00 00 00 00 00 00
.:2B88 00 00 00 00 00 00 00 00
.:2B90 00 00 00 00 00 00 00 00
.:2B98 00 00 00 00 00 00 00 00
.:2BA0 00 00 00 00 00 00 00 00
.:2BA8 00 00 00 00 00 00 00 00
.:2BB0 00 00 00 00 00 00 00 00
.:2BB8 00 00 00 00 00 00 00 00
.:2BC0 00 00 00 00 00 00 00 00
.:2BC8 00 00 00 00 00 00 00 00
.:2BD0 00 00 00 00 00 00 00 00
.:2BD8 00 00 00 00 00 00 00 00
.:2BE0 00 00 00 00 00 00 00 00
.:2BE8 00 00 00 00 00 00 00 00
.:2BF0 00 00 00 00 00 00 00 00
.:2BF8 00 00 00 00 00 00 00 00
.:2C00 00 00 00 00 00 00 00 00
.:2C08 00 00 00 00 00 00 00 00
.:2C10 00 00 00 00 00 00 00 00
.:2C18 00 00 00 00 00 00 00 00
.:2C20 00 00 00 00 00 00 00 00
.:2C28 00 00 00 00 00 00 00 00
.:2C30 00 00 00 00 00 00 00 00
.:2C38 00 00 00 00 00 00 00 00
.:2C40 00 00 00 00 00 00 00 00
.:2C48 00 00 00 00 00 00 00 00
.:2C50 00 00 00 00 00 00 00 00
.:2C58 00 00 00 00 00 00 00 00
.:2C60 00 00 00 00 00 00 00 00
.:2C68 00 00 00 00 00 00 00 00
.:2C70 00 00 00 00 00 00 00 00
.:2C78 00 00 00 00 00 00 00 00
.:2C80 3F 30 30 30 30 30 30 30
.:2C88 FF 00 00 00 00 00 00 00
.:2C90 FF 00 00 00 00 00 00 00
.:2C98 FF 00 00 00 00 00 00 00
.:2CA0 FF 00 00 00 00 00 00 00
.:2CA8 FF 00 00 00 00 00 00 00
.:2CB0 FF 00 00 00 00 00 00 00
.:2CB8 FF 00 00 00 00 00 00 00
.:2CC0 FF 00 00 00 00 00 00 00
.:2CC8 FF 00 00 00 00 00 00 00
.:2CD0 FF 00 00 00 00 00 00 00
.:2CD8 FF 00 00 00 00 00 00 00
.:2CE0 FF 00 00 00 00 00 00 00
.:2CE8 FF 00 00 00 00 00 00 00
.:2CF0 FF 00 00 00 00 00 00 00
.:2CF8 FF 00 00 00 00 00 00 00
.:2D00 FF 00 00 00 00 00 00 00
.:2D08 FF 00 00 00 00 00 00 00
.:2D10 FF 00 00 00 00 00 00 00
.:2D18 FF 00 00 00 00 00 00 00
.:2D20 FF 00 00 00 00 00 00 00
.:2D28 FF 00 00 00 00 00 00 00
.:2D30 FF 00 00 00 00 00 00 00
.:2D38 FF 00 00 00 00 00 00 00
.:2D40 FF 00 00 00 00 00 00 00
.:2D48 FF 00 00 00 00 00 00 00
.:2D50 FF 00 00 00 00 00 00 00
.:2D58 FF 00 00 00 00 00 00 00
.:2D60 FF 00 00 00 00 00 00 00
.:2D68 FF 00 00 00 00 00 00 00
.:2D70 FF 00 00 00 00 00 00 00
.:2D78 FF 00 00 00 00 00 00 00
.:2D80 FF 00 00 00 00 00 00 00
.:2D88 FF 00 00 00 00 00 00 00
.:2D90 FF 00 00 00 00 00 00 00
.:2D98 FF 00 00 00 00 00 00 00
```

```
.:2DA0 FF 00 00 00 00 00 00 00
.:2DA8 FF 00 00 00 00 00 00 00
.:2DB0 FF 00 00 00 00 00 00 00
.:2DB8 FC 0C 0C 0C 0C 0C 0C 0C
.:2DC0 30 30 30 30 30 30 33 33
.:2DC8 00 00 00 00 00 00 03 03
.:2DD0 00 00 00 00 00 00 03 0C
.:2DD8 00 00 00 00 00 00 F0 0C
.:2DE0 00 00 00 00 00 00 30 30
.:2DE8 00 00 00 00 00 00 30 30
.:2DF0 00 00 00 00 00 00 FF C0
.:2DF8 00 00 00 00 00 00 00 C0
.:2E00 00 00 00 00 00 00 00 00
.:2E08 00 00 00 00 00 00 03 03
.:2E10 00 00 00 00 00 00 00 00
.:2E18 00 00 00 00 00 00 03 0C
.:2E20 00 00 00 00 00 00 F0 0C
.:2E28 00 00 00 00 00 00 0F 30
.:2E30 00 00 00 00 00 00 C0 30
.:2E38 00 00 00 00 00 00 3F C0
.:2E40 00 00 00 00 00 00 00 C0
.:2E48 00 00 00 00 00 00 00 00
.:2E50 00 00 00 00 00 00 00 00
.:2E58 00 00 00 00 00 00 03 03
.:2E60 00 00 00 00 00 00 03 03
.:2E68 00 00 00 00 00 00 0F 0C
.:2E70 00 00 00 00 00 00 FC 00
.:2E78 00 00 00 00 00 00 3F 30
.:2E80 00 00 00 00 00 00 C0 30
.:2E88 00 00 00 00 00 00 FF C0
.:2E90 00 00 00 00 00 00 C0 00
.:2E98 00 00 00 00 00 00 00 00
.:2EA0 00 00 00 00 00 00 00 00
.:2EA8 00 00 00 00 00 00 00 00
.:2EB0 00 00 00 00 00 00 00 00
.:2EB8 00 00 00 00 00 00 00 00
.:2EC0 00 00 00 00 00 00 00 00
.:2EC8 00 00 00 00 00 00 00 00
.:2ED0 00 00 00 00 00 00 00 00
.:2ED8 00 00 00 00 00 00 00 00
.:2EE0 00 00 00 00 00 00 00 00
.:2EE8 00 00 00 00 00 00 00 00
.:2EF0 00 00 00 00 00 00 00 00
.:2EF8 0C 0C 0C 0C 0C 0C 0C 0C
.:2F00 33 30 30 30 30 30 30 30
.:2F08 03 CC CC CC 30 30 30 30
.:2F10 0C 0C 0C 0C 0C 0C 0C 0C
.:2F18 0C 0C 0C 0C 0C 0C 0C 0C
.:2F20 30 30 30 30 30 30 30 30
.:2F28 30 30 30 30 30 30 30 30
.:2F30 C0 C0 C0 FF CC C3 C3 C0
.:2F38 C0 C0 C0 00 00 00 00 C0
.:2F40 00 00 00 00 00 00 00 00
.:2F48 03 03 03 03 03 03 03 03
.:2F50 00 00 00 00 00 00 00 00
.:2F58 0C 0C 0C 0C 0C 0C 0C 0C
.:2F60 0C 0C 0C 0C 0C 0C 0C 0C
.:2F68 30 30 30 30 30 30 30 30
.:2F70 30 30 00 00 F0 30 30 30
.:2F78 C0 C0 C0 C0 C0 C0 C0 C0
.:2F80 C0 C0 C0 C0 C0 C0 C0 C0
.:2F88 00 00 00 00 00 00 00 00
.:2F90 00 00 00 00 00 00 00 00
.:2F98 03 03 03 03 03 03 03 03
.:2FA0 03 03 03 03 FF 03 03 03
.:2FA8 0C 0C 0C 0C 0F 0C 0C 0C
.:2FB0 00 00 00 00 F0 00 00 00
.:2FB8 30 30 30 3F 33 30 30 30
.:2FC0 30 30 30 C0 00 C0 C0 30
.:2FC8 C0 C0 C0 C0 FF C0 C0 C0
.:2FD0 00 00 00 00 00 00 00 00
.:2FD8 00 00 00 00 00 00 00 00
.:2FE0 00 00 00 00 00 00 00 00
.:2FE8 00 00 00 00 00 00 00 00
.:2FF0 00 00 00 00 00 00 00 00
.:2FF8 00 00 00 00 00 00 00 00
.:3000 00 00 00 00 00 00 00 00
.:3008 00 00 00 00 00 00 00 00
.:3010 00 00 00 00 00 00 00 00
.:3018 00 00 00 00 00 00 00 00
.:3020 00 00 00 00 00 00 00 00
.:3028 00 00 00 00 00 00 00 00
.:3030 00 00 00 00 00 00 00 00
.:3038 0C 0C 0C 0C 0C 0C 0C 0C
.:3040 30 30 30 30 30 30 30 30
.:3048 30 30 30 00 00 00 00 00
.:3050 0C 0C 03 00 00 00 00 00
.:3058 0C 0C F0 00 00 00 00 00
.:3060 30 30 0F 00 00 00 00 00
.:3068 30 30 C0 00 00 00 00 00
.:3070 C0 C0 C0 00 00 00 00 00
.:3078 C0 C0 C0 00 00 00 00 00
.:3080 00 00 00 00 00 00 00 00
.:3088 03 03 03 00 00 00 00 00
.:3090 00 00 FF 00 00 00 00 00
.:3098 0C 0C 03 00 00 00 00 00
.:30A0 0C 0C F0 00 00 00 00 00
.:30A8 30 30 0F 00 00 00 00 00
.:30B0 30 30 C0 00 00 00 00 00
.:30B8 C0 C0 3F 00 00 00 00 00
.:30C0 C0 C0 00 00 00 00 00 00
.:30C8 00 00 00 00 00 00 00 00
.:30D0 00 00 00 00 00 00 00 00
.:30D8 03 03 03 00 00 00 00 00
.:30E0 03 03 03 00 00 00 00 00
.:30E8 0C 0C 0F 00 00 00 00 00
.:30F0 00 00 FC 00 00 00 00 00
.:30F8 30 30 30 00 00 00 00 00
.:3100 30 30 30 00 00 00 00 00
.:3108 C0 C0 FF 00 00 00 00 00
.:3110 00 00 C0 00 00 00 00 00
.:3118 00 00 0C 00 00 00 00 00
.:3120 00 00 03 00 00 00 00 00
.:3128 00 00 00 00 00 00 00 00
.:3130 00 00 C0 00 00 00 00 00
.:3138 00 00 0C 00 00 00 00 00
.:3140 00 00 00 00 00 00 00 00
.:3148 00 00 00 00 00 00 00 00
```

```
.:3150 00 00 00 00 00 00 00 00
.:3158 00 00 00 00 00 00 00 00
.:3160 00 00 00 00 00 00 00 00
.:3168 00 00 00 00 00 00 00 00
.:3170 00 00 00 00 00 00 00 00
.:3178 0C 0C 0C 0C 0C 0C 0C 0C
.:3180 30 30 30 30 30 30 30 30
.:3188 00 00 00 00 00 00 00 00
.:3190 00 00 00 00 00 00 00 00
.:3198 00 00 00 00 00 00 00 00
.:31A0 00 00 00 00 00 00 00 00
.:31A8 00 00 00 00 00 00 00 00
.:31B0 00 00 00 00 00 00 00 00
.:31B8 00 00 00 00 00 00 00 00
.:31C0 00 00 00 00 00 00 00 00
.:31C8 00 00 00 00 00 00 00 00
.:31D0 00 00 00 00 00 00 00 00
.:31D8 00 00 00 00 00 00 00 00
.:31E0 00 00 00 00 00 00 00 00
.:31E8 00 00 00 00 00 00 00 00
.:31F0 00 00 00 00 00 00 00 00
.:31F8 00 00 00 00 00 00 00 00
.:3200 00 00 00 00 00 00 00 00
.:3208 00 00 00 00 00 00 00 00
.:3210 00 00 00 00 00 00 00 00
.:3218 00 00 00 00 00 00 00 00
.:3220 00 00 00 00 00 00 00 00
.:3228 00 00 00 00 00 00 00 00
.:3230 00 00 00 00 00 00 00 00
.:3238 00 00 00 00 00 00 00 00
.:3240 00 00 00 00 00 00 00 00
.:3248 00 00 00 00 00 00 00 00
.:3250 00 00 00 00 00 00 00 00
.:3258 00 00 00 00 00 00 00 00
.:3260 00 00 00 00 00 00 00 00
.:3268 00 00 00 00 00 00 00 00
.:3270 00 00 00 00 00 00 00 00
.:3278 00 00 00 00 00 00 00 00
.:3280 00 00 00 00 00 00 00 00
.:3288 00 00 00 00 00 00 00 00
.:3290 00 00 00 00 00 00 00 00
.:3298 00 00 00 00 00 00 00 00
.:32A0 00 00 00 00 00 00 00 00
.:32A8 00 00 00 00 00 00 00 00
.:32B0 00 00 00 00 00 00 00 00
.:32B8 0C 0C 0C 0C 0C 0C 0C 0C
.:32C0 30 30 30 30 30 30 30 30
.:32C8 00 00 00 00 00 00 00 00
.:32D0 00 00 00 00 00 00 00 00
.:32D8 00 00 00 00 00 00 00 00
.:32E0 00 00 00 00 00 00 00 00
.:32E8 00 00 00 00 00 00 00 00
.:32F0 00 00 00 00 00 00 00 00
.:32F8 00 00 00 00 00 00 00 00
.:3300 00 00 00 00 00 00 00 00
.:3308 00 00 00 00 00 00 00 00
.:3310 00 00 00 00 00 00 00 00
.:3318 00 00 00 00 00 00 00 00
.:3320 00 00 00 00 00 00 00 00
.:3328 00 00 00 00 00 00 00 00
.:3330 00 00 00 00 00 00 00 00
.:3338 00 00 00 00 00 00 00 00
.:3340 00 00 00 00 00 00 00 00
.:3348 00 00 00 00 00 00 00 00
.:3350 00 00 00 00 00 00 00 00
.:3358 00 00 00 00 00 00 00 00
.:3360 00 00 00 00 00 00 00 00
.:3368 00 00 00 00 00 00 00 00
.:3370 00 00 00 00 00 00 00 00
.:3378 00 00 00 00 00 00 00 00
.:3380 00 00 00 00 00 00 00 00
.:3388 00 00 00 00 00 00 00 00
.:3390 00 00 00 00 00 00 00 00
.:3398 00 00 00 00 00 00 00 00
.:33A0 00 00 00 00 00 00 00 00
.:33A8 00 00 00 00 00 00 00 00
.:33B0 00 00 00 00 00 00 00 00
.:33B8 00 00 00 00 00 00 00 00
.:33C0 00 00 00 00 00 00 00 00
.:33C8 00 00 00 00 00 00 00 00
.:33D0 00 00 00 00 00 00 00 00
.:33D8 00 00 00 00 00 00 00 00
.:33E0 00 00 00 00 00 00 00 00
.:33E8 00 00 00 00 00 00 00 00
.:33F0 00 00 00 00 00 00 00 00
.:33F8 0C 0C 0C 0C 0C 0C 0C 0C
.:3400 30 30 30 30 30 30 30 30
.:3408 00 00 00 00 00 00 00 00
.:3410 00 00 00 00 00 00 00 00
.:3418 00 00 00 00 00 00 00 00
.:3420 00 00 00 00 00 00 00 00
.:3428 00 00 00 00 00 00 00 00
.:3430 00 00 00 00 00 00 00 00
.:3438 00 00 00 00 00 00 00 00
.:3440 00 00 00 00 00 00 00 00
.:3448 00 00 00 00 00 00 00 00
.:3450 00 00 00 00 00 00 00 00
.:3458 00 00 00 00 00 00 00 00
.:3460 00 00 00 00 00 00 00 00
.:3468 00 00 00 00 00 00 00 00
.:3470 00 00 00 00 00 00 00 00
.:3478 00 00 00 00 00 00 00 00
.:3480 00 00 00 00 00 00 00 00
.:3488 00 00 00 00 00 00 00 00
.:3490 00 00 00 00 00 00 00 00
.:3498 00 00 00 00 00 00 00 00
.:34A0 00 00 00 00 00 00 00 00
.:34A8 00 00 00 00 00 00 00 00
.:34B0 00 00 00 00 00 00 00 00
.:34B8 00 00 00 00 00 00 00 00
.:34C0 00 00 00 00 00 00 00 00
.:34C8 00 00 00 00 00 00 00 00
.:34D0 00 00 00 00 00 00 00 00
.:34D8 00 00 00 00 00 00 00 00
.:34E0 00 00 00 00 00 00 00 00
.:34E8 00 00 00 00 00 00 00 00
.:34F0 00 00 00 00 00 00 00 00
.:34F8 00 00 00 00 00 00 00 00
```

```
.:3500 00 00 00 00 00 00 00 00
.:3508 00 00 00 00 00 00 00 00
.:3510 00 00 00 00 00 00 00 00
.:3518 00 00 00 00 00 00 00 00
.:3520 00 00 00 00 00 00 00 00
.:3528 00 00 00 00 00 00 00 00
.:3530 00 00 00 00 00 00 00 00
.:3538 0C 0C 0C 0C 0C 0C 0C 0C
.:3540 30 30 30 3F 00 00 00 00
.:3548 00 00 00 FF 00 00 00 00
.:3550 00 00 00 FF 00 00 00 00
.:3558 00 00 00 FF 00 00 00 00
.:3560 00 00 00 FF 00 00 00 00
.:3568 00 00 00 FF 00 00 00 00
.:3570 00 00 00 FF 00 00 00 00
.:3578 00 00 00 FF 00 00 00 00
.:3580 00 00 00 FF 00 00 00 00
.:3588 00 00 00 FF 00 00 00 00
.:3590 00 00 00 FF 00 00 00 00
.:3598 00 00 00 FF 00 00 00 00
.:35A0 00 00 00 FF 00 00 00 00
.:35A8 00 00 00 FF 00 00 00 00
.:35B0 00 00 00 FF 00 00 00 00
.:35B8 00 00 00 FF 00 00 00 00
.:35C0 00 00 00 FF 00 00 00 00
.:35C8 00 00 00 FF 00 00 00 00
.:35D0 00 00 00 FF 00 00 00 00
.:35D8 00 00 00 FF 00 00 00 00
.:35E0 00 00 00 FF 00 00 00 00
.:35E8 00 00 00 FF 00 00 00 00
.:35F0 00 00 00 FF 00 00 00 00
.:35F8 00 00 00 FF 00 00 00 00
.:3600 00 00 00 FF 00 00 00 00
.:3608 00 00 00 FF 00 00 00 00
.:3610 00 00 00 FF 00 00 00 00
.:3618 00 00 00 FF 00 00 00 00
.:3620 00 00 00 FF 00 00 00 00
.:3628 00 00 00 FF 00 00 00 00
.:3630 00 00 00 FF 00 00 00 00
.:3638 00 00 00 FF 00 00 00 00
.:3640 00 00 00 FF 00 00 00 00
.:3648 00 00 00 FF 00 00 00 00
.:3650 00 00 00 FF 00 00 00 00
.:3658 00 00 00 FF 00 00 00 00
.:3660 00 00 00 FF 00 00 00 00
.:3668 00 00 00 FF 00 00 00 00
.:3670 00 00 00 FF 00 00 00 00
.:3678 0C 0C 0C FC 00 00 00 00
.:3680 00 00 00 00 00 00 00 00
.:3688 00 00 00 00 00 00 00 00
.:3690 00 00 00 00 00 28 82 02
.:3698 00 00 00 00 00 00 00 00
.:36A0 00 00 00 00 00 A0 88 A0
.:36A8 00 00 00 00 00 80 80 80
.:36B0 00 00 00 00 00 20 20 88
.:36B8 00 00 00 00 00 88 88 A8
.:36C0 00 00 00 00 00 A8 80 80
.:36C8 00 00 00 00 00 A0 88 A0
.:36D0 00 00 00 00 00 28 80 80
.:36D8 00 00 00 00 00 00 00 00
.:36E0 00 00 00 00 00 00 00 00
.:36E8 00 00 00 00 00 00 00 00
.:36F0 00 00 00 00 00 00 00 00
.:36F8 00 00 00 00 00 11 11 11
.:3700 00 00 00 00 00 10 10 10
.:3708 00 00 00 00 00 20 20 22
.:3710 00 00 00 00 00 22 22 22
.:3718 00 00 00 00 00 2A 02 08
.:3720 00 00 00 00 00 04 04 11
.:3728 00 00 00 00 00 14 11 14
.:3730 00 00 00 00 00 28 22 22
.:3738 00 00 00 00 00 00 00 00
.:3740 00 00 00 00 00 00 00 00
.:3748 00 00 00 00 00 00 00 00
.:3750 00 00 00 00 00 00 00 00
.:3758 00 00 00 00 00 00 00 00
.:3760 00 00 00 00 00 08 08 08
.:3768 00 00 00 00 00 02 08 08
.:3770 00 00 00 00 00 02 82 82
.:3778 00 00 00 00 00 02 02 02
.:3780 00 00 00 00 00 22 20 20
.:3788 00 00 00 00 00 A0 20 82
.:3790 00 00 00 00 00 82 82 22
.:3798 00 00 00 00 00 82 22 82
.:37A0 00 00 00 00 00 80 20 20
.:37A8 00 00 00 00 00 00 00 00
.:37B0 00 00 00 00 00 00 00 00
.:37B8 00 00 00 00 00 00 00 00
.:37C0 00 00 00 00 00 00 00 00
.:37C8 00 00 00 00 00 00 00 00
.:37D0 08 20 80 AA 00 00 00 00
.:37D8 00 00 00 00 00 00 00 00
.:37E0 80 80 80 80 00 00 00 00
.:37E8 80 80 80 A8 00 00 00 00
.:37F0 88 A8 88 88 00 00 00 00
.:37F8 20 20 20 20 00 00 00 00
.:3800 A0 80 80 A8 00 00 00 00
.:3808 88 88 88 88 00 00 00 00
.:3810 20 08 08 A0 00 00 00 00
.:3818 00 00 00 00 00 00 00 00
.:3820 00 00 00 00 00 00 00 00
.:3828 00 00 00 00 00 00 00 00
.:3830 00 00 00 00 00 00 00 00
.:3838 2A 22 22 22 00 00 00 00
.:3840 20 20 20 20 00 00 00 00
.:3848 22 2A 28 08 00 00 00 00
.:3850 22 A2 A2 82 00 00 00 00
.:3858 08 08 20 2A 00 00 00 00
.:3860 22 2A 22 22 00 00 00 00
.:3868 22 22 22 22 00 00 00 00
.:3870 22 22 22 28 00 00 00 00
.:3878 00 00 00 00 00 00 00 00
.:3880 00 00 00 00 00 00 00 00
.:3888 00 00 00 00 00 00 00 00
.:3890 00 00 00 00 00 00 00 00
.:3898 00 00 00 00 00 00 00 00
.:38A0 08 08 08 0A 00 00 00 00
.:38A8 08 08 08 82 00 00 00 00
```

```
.:38B0 82 82 82 00 00 00 00 00
.:38B8 22 AA 8A 88 00 00 00 00
.:38C0 20 20 22 22 00 00 00 00
.:38C8 82 82 02 A2 00 00 00 00
.:38D0 22 A2 22 22 00 00 00 00
.:38D8 22 22 22 22 00 00 00 00
.:38E0 20 20 20 80 00 00 00 00
.:38E8 00 00 00 00 00 00 00 00
.:38F0 00 00 00 00 00 00 00 00
.:38F8 00 00 00 00 00 00 00 00
.:3900 00 00 00 00 00 00 00 00
.:3908 00 00 00 00 00 00 00 00
.:3910 00 00 00 00 00 00 00 00
.:3918 00 CC CC FF 15 11 11 0C
.:3920 00 C0 C2 C2 02 02 02 02
.:3928 00 00 00 00 00 00 00 00
.:3930 00 44 44 55 3F 33 33 08
.:3938 00 C0 C2 C2 02 02 02 02
.:3940 00 00 00 00 00 00 00 00
.:3948 00 00 00 00 00 00 00 00
.:3950 00 00 00 00 00 00 00 00
.:3958 00 00 00 00 00 00 00 00
.:3960 00 00 00 00 00 00 00 00
.:3968 00 00 00 00 00 00 00 00
.:3970 00 00 00 00 00 00 00 00
.:3978 00 00 00 00 00 00 00 00
.:3980 00 00 00 00 00 00 00 00
.:3988 00 00 00 00 00 00 00 00
.:3990 00 0C 0C 0F 02 02 02 00
.:3998 00 CC CC FC A0 20 20 40
.:39A0 00 00 30 30 30 30 30 30
.:39A8 00 00 00 00 00 00 00 00
.:39B0 00 00 00 00 00 00 00 00
.:39B8 00 00 00 00 00 00 00 00
.:39C0 00 00 00 00 00 00 00 00
.:39C8 00 00 00 00 00 00 00 00
.:39D0 00 00 00 00 00 00 00 00
.:39D8 00 00 00 00 00 00 00 00
.:39E0 00 00 00 00 00 00 00 00
.:39E8 00 00 00 00 00 00 00 00
.:39F0 00 00 00 00 00 00 00 00
.:39F8 00 0C 0C 0F 02 02 02 00
.:3A00 00 CC CC FC A0 20 20 40
.:3A08 00 00 30 30 30 30 30 30
.:3A10 00 00 00 00 00 00 00 00
.:3A18 00 00 00 00 00 00 00 00
.:3A20 00 00 00 00 00 00 00 00
.:3A28 00 00 00 00 00 00 00 00
.:3A30 00 00 00 00 00 00 00 00
.:3A38 00 00 00 00 00 00 00 00
.:3A40 00 00 00 00 00 00 00 00
.:3A48 00 00 00 00 00 00 00 00
.:3A50 00 00 00 02 00 00 00 00
.:3A58 0C 1F 5F 55 CC CC FF FF
.:3A60 03 03 87 57 83 83 83 83
.:3A68 00 00 00 03 00 00 00 00
.:3A70 08 3A FA FF 88 88 AA AA
.:3A78 01 01 8D FD 81 81 81 81
.:3A80 00 00 00 00 00 00 00 00
.:3A88 00 00 00 00 00 00 00 00
.:3A90 00 00 00 00 00 00 00 00
.:3A98 00 00 00 00 00 00 00 00
.:3AA0 00 00 00 00 00 00 00 00
.:3AA8 00 00 00 00 00 00 00 00
.:3AB0 00 00 00 00 00 00 00 00
.:3AB8 00 00 00 00 00 00 00 00
.:3AC0 00 00 00 00 00 00 00 00
.:3AC8 00 00 00 00 00 00 00 00
.:3AD0 00 03 0F 3F 08 08 0A 03
.:3AD8 80 A0 A8 FF 88 88 A8 30
.:3AE0 30 30 B0 B0 30 30 30 30
.:3AE8 00 00 00 00 00 00 00 00
.:3AF0 00 00 00 00 00 00 00 00
.:3AF8 00 00 00 00 00 00 00 00
.:3B00 00 00 00 00 00 00 00 00
.:3B08 00 00 00 00 00 00 00 00
.:3B10 00 00 00 00 00 00 00 00
.:3B18 00 00 00 00 00 00 00 00
.:3B20 00 00 00 00 00 00 00 00
.:3B28 00 00 00 00 00 00 00 00
.:3B30 00 00 00 00 00 00 00 00
.:3B38 00 03 0F 3F 08 08 0A 03
.:3B40 80 A0 A8 FF 88 88 A8 30
.:3B48 30 30 B0 B0 30 30 30 30
.:3B50 00 00 00 00 00 00 00 00
.:3B58 00 00 00 00 00 00 00 00
.:3B60 00 00 00 00 00 00 00 00
.:3B68 00 00 00 00 00 00 00 00
.:3B70 00 00 00 00 00 00 00 00
.:3B78 00 00 00 00 00 00 00 00
.:3B80 00 00 00 00 00 00 00 00
.:3B88 00 00 00 00 00 00 00 00
.:3B90 00 00 00 00 00 00 00 00
.:3B98 11 11 11 1D 00 00 00 00
.:3BA0 02 00 00 C0 00 00 00 00
.:3BA8 00 00 00 00 00 00 00 00
.:3BB0 33 33 33 3B 00 00 00 00
.:3BB8 02 00 00 C0 00 00 00 00
.:3BC0 00 00 00 00 00 00 00 00
.:3BC8 00 00 00 00 00 00 00 00
.:3BD0 00 00 00 00 00 00 00 00
.:3BD8 00 00 00 00 00 00 00 00
.:3BE0 00 00 00 00 00 00 00 00
.:3BE8 00 00 00 00 00 00 00 00
.:3BF0 00 00 00 00 00 00 00 00
.:3BF8 00 00 00 00 00 00 00 00
.:3C00 00 00 00 00 00 00 00 00
.:3C08 00 00 00 00 00 00 00 00
.:3C10 02 02 02 00 00 00 00 00
.:3C18 20 20 EC 00 00 00 00 00
.:3C20 00 00 00 00 00 00 00 00
.:3C28 00 00 00 00 00 00 00 00
.:3C30 00 00 00 00 00 00 00 00
.:3C38 00 00 00 00 00 00 00 00
.:3C40 00 00 00 00 00 00 00 00
.:3C48 00 00 00 00 00 00 00 00
.:3C50 00 00 00 00 00 00 00 00
.:3C58 00 00 00 00 00 00 00 00
```

```
.:3C60 00 00 00 00 00 00 00 00
.:3C68 00 00 00 00 00 00 00 00
.:3C70 00 00 00 00 00 00 00 00
.:3C78 02 02 02 02 00 00 00 00
.:3C80 20 20 20 EC 00 00 00 00
.:3C88 20 00 00 00 00 00 00 00
.:3C90 00 00 00 00 00 00 00 00
.:3C98 00 00 00 00 00 00 00 00
.:3CA0 00 00 00 00 00 00 00 00
.:3CA8 00 00 00 00 00 00 00 00
.:3CB0 00 00 00 00 00 00 00 00
.:3CB8 00 00 00 00 00 00 00 00
.:3CC0 00 00 00 00 00 00 00 00
.:3CC8 00 00 00 00 0A 20 80 8A
.:3CD0 00 00 00 00 00 80 20 20
.:3CD8 00 00 00 00 00 00 02 02
.:3CE0 00 00 00 00 28 A2 80 00
.:3CE8 00 00 00 00 00 02 08 08
.:3CF0 00 00 00 00 A0 08 02 02
.:3CF8 00 00 00 00 0A 08 08 08
.:3D00 00 00 00 00 80 20 08 08
.:3D08 00 00 00 00 20 20 08 08
.:3D10 00 00 00 00 20 20 80 80
.:3D18 00 00 00 00 A8 82 80 80
.:3D20 00 00 00 00 02 00 80 80
.:3D28 00 00 00 00 A0 80 80 80
.:3D30 00 00 00 00 0A 20 80 80
.:3D38 00 00 00 00 02 82 82 02
.:3D40 00 00 00 00 08 08 08 08
.:3D48 00 00 00 00 2A 02 02 02
.:3D50 00 00 00 00 A0 00 00 00
.:3D58 00 00 00 00 02 0A 02 02
.:3D60 00 00 00 00 02 08 08 08
.:3D68 00 00 00 00 80 20 20 20
.:3D70 00 00 00 00 28 82 82 82
.:3D78 00 00 00 00 02 02 08 08
.:3D80 00 00 00 00 80 80 80 80
.:3D88 00 00 00 00 00 00 00 00
.:3D90 00 00 00 00 2A 20 20 20
.:3D98 00 00 00 00 82 02 02 02
.:3DA0 00 00 00 00 02 82 82 82
.:3DA8 00 00 00 00 0A 00 00 00
.:3DB0 00 00 00 00 A8 80 80 80
.:3DB8 00 00 00 00 2A 20 20 20
.:3DC0 00 00 00 00 00 80 82 82
.:3DC8 00 00 00 00 20 88 02 02
.:3DD0 00 00 00 00 0F 0C 0C 0C
.:3DD8 00 00 00 00 80 20 20 20
.:3DE0 00 00 00 00 80 80 22 22
.:3DE8 00 00 00 00 80 80 00 00
.:3DF0 00 00 00 00 00 00 00 00
.:3DF8 00 00 00 00 00 00 00 00
.:3E00 00 00 00 00 00 00 00 00
.:3E08 88 88 8A 80 20 0A 00 00
.:3E10 20 20 20 20 80 00 00 00
.:3E18 02 02 02 02 00 00 00 00
.:3E20 00 00 00 80 A2 28 00 00
.:3E28 08 08 08 08 02 00 00 00
.:3E30 02 02 02 02 08 A0 00 00
.:3E38 08 0A 08 08 08 08 00 00
.:3E40 20 80 00 00 00 00 00 00
.:3E48 02 02 02 02 02 02 00 00
.:3E50 00 00 00 00 00 00 00 00
.:3E58 82 A8 88 82 80 80 00 00
.:3E60 00 00 00 00 80 82 00 00
.:3E68 80 80 80 80 80 A0 00 00
.:3E70 80 82 80 80 20 0A 00 00
.:3E78 02 82 82 82 82 02 00 00
.:3E80 A8 08 08 08 08 08 00 00
.:3E88 02 02 02 02 02 02 00 00
.:3E90 00 00 00 00 00 00 00 00
.:3E98 02 02 02 02 02 0A 00 00
.:3EA0 02 00 00 00 02 80 00 00
.:3EA8 A0 20 20 20 20 80 00 00
.:3EB0 28 28 82 82 82 28 00 00
.:3EB8 0A 00 00 00 00 00 00 00
.:3EC0 A0 80 80 80 80 80 00 02
.:3EC8 00 00 00 00 80 80 80 00
.:3ED0 2A 20 20 20 20 2A 00 00
.:3ED8 02 02 02 02 02 82 00 00
.:3EE0 22 22 0A 0A 0A 02 00 00
.:3EE8 00 00 00 00 00 00 00 00
.:3EF0 80 80 80 80 80 80 00 00
.:3EF8 2A 22 20 20 20 20 00 00
.:3F00 02 02 82 82 80 80 00 00
.:3F08 02 02 02 02 88 20 00 00
.:3F10 0A 08 08 08 08 08 00 00
.:3F18 80 00 00 00 00 00 00 00
.:3F20 08 08 08 08 08 08 00 00
.:3F28 00 00 00 00 00 00 00 00
.:3F30 00 00 00 00 00 00 00 00
.:3F38 00 00 00 00 00 00 00 00
.:3F40 3D DF 7D BB 7F EF 7D 7F
.:3F48 7F EE 7D 8F 7F CE 7D FF
.:3F50 3F 9E 7F 82 7F 80 7F 8D
.:3F58 3F 82 7F C0 3D C7 7D DF
.:3F60 7F FF 7F 90 7D 8F 3D CF
.:3F68 7D 80 7F 8B 7F 82 7D 04
.:3F70 7F 86 3F 92 3D 82 7F CB
.:3F78 7D 8A 7D BE BD BA 7D 00
.:3F80 9B F5 8F F5 FF B1 4A B0
.:3F88 8B F5 0E B1 9A 31 4B 31
.:3F90 DB 75 0A F5 8A F5 0A F1
.:3F98 CE F5 0A F5 DB F1 7F 71
.:3FA0 8A 71 0A F1 CF F1 0A F1
.:3FA8 AA F5 FB F1 9B F5 4B 71
.:3FB0 FF 71 FF F5 DF F5 4B F5
.:3FB8 DF F5 3B 71 FF F5 FF 71
.:3FC0 8C 7D 8C 75 B4 75 80 8A
.:3FC8 80 79 84 79 D0 39 81 31
.:3FD0 D8 79 B0 7D 80 7D 80 75
.:3FD8 C6 7D 90 7D C5 79 F5 31
.:3FE0 80 79 82 7D BA 75 C7 71
.:3FE8 AA 7D C2 75 C5 7D 8A FB
.:3FF0 F7 79 FF 7D D2 7D 8A 7D
.:3FF8 97 75 F3 71 FF 7D FF FF
.:4000 68 00 29 08 14 20 08 C4
.:4008 20 68 04 29 09 66 60 08
```

```
.:4010 2A 20 68 26 29 00 A6 80
.:4018 02 66 60 2A B3 AA 11 33
.:4020 11 00 33 00 00 FF C0 00
.:4028 CC C0 00 00 00 00 00 00
.:4030 00 00 00 00 00 00 00 00
.:4038 00 00 00 00 00 00 00 00
.:4040 68 00 29 08 68 20 29 C4
.:4048 29 68 04 29 69 66 69 68
.:4050 2A 29 68 26 29 00 A6 80
.:4058 02 66 60 2A B3 AA 11 33
.:4060 11 00 33 00 00 FF C0 00
.:4068 CC C0 00 00 00 00 00 00
.:4070 00 00 00 00 00 00 00 00
.:4078 00 00 00 00 00 00 00 00
.:4080 68 00 29 08 68 20 29 C4
.:4088 29 68 04 29 69 66 69 68
.:4090 2A 29 68 26 29 00 A6 80
.:4098 02 66 60 2A B3 AA 11 33
.:40A0 11 00 33 00 00 FF C0 00
.:40A8 CC C0 05 55 54 05 55 54
.:40B0 00 00 00 00 00 00 00 00
.:40B8 00 00 00 00 00 00 00 00
.:40C0 68 00 29 08 14 20 08 C4
.:40C8 20 68 04 29 09 66 60 08
.:40D0 2A 20 68 26 29 00 A6 80
.:40D8 02 66 60 2A B3 AA 11 33
.:40E0 11 00 33 00 00 FF C0 00
.:40E8 CC C0 05 55 54 05 55 54
.:40F0 00 00 00 00 00 00 00 00
.:40F8 00 00 00 00 00 00 00 00
.:4100 11 11 00 15 51 00 02 81
.:4108 00 0A 81 00 03 01 00 07
.:4110 C1 00 17 C1 00 5F C1 00
.:4118 15 5A 00 3F F1 00 37 71
.:4120 00 3F F1 00 04 40 00 04
.:4128 40 00 07 70 00 00 00 00
.:4130 00 00 00 00 00 00 00 00
.:4138 00 00 00 00 00 00 00 00
.:4140 11 11 00 15 51 00 02 81
.:4148 00 0A 81 00 03 01 00 07
.:4150 C1 00 17 C1 00 5F C1 00
.:4158 15 5A 00 3F F1 00 37 71
.:4160 00 3F F1 00 04 40 00 04
.:4168 40 00 07 70 00 00 00 00
.:4170 00 00 00 00 00 00 00 00
.:4178 00 00 00 00 00 00 00 00
.:4180 08 00 00 20 00 00 28 00
.:4188 00 02 00 00 02 00 00 08
.:4190 00 00 00 00 00 00 00 00
.:4198 00 00 00 00 00 00 00 00
.:41A0 00 00 00 00 00 00 00 00
.:41A8 00 00 00 00 00 00 00 00
.:41B0 00 00 00 00 00 00 00 00
.:41B8 00 00 00 00 00 00 00 00
.:41C0 08 00 00 02 00 00 02 00
.:41C8 00 28 00 00 20 00 00 08
.:41D0 00 00 00 00 00 00 00 00
.:41D8 00 00 00 00 00 00 00 00
.:41E0 00 00 00 00 00 00 00 00
.:41E8 00 00 00 00 00 00 00 00
.:41F0 00 00 00 00 00 00 00 00
.:41F8 00 00 00 00 00 00 00 00
.:4200 FF FF FF FF FF FF FF FF
.:4208 FF FF FF FF FF FF FF FF
.:4210 FF FF FF FF FF FF FF FF
.:4218 FF FF FF FF FF FF FF FF
.:4220 FF FF FF FF FF FF FF FF
.:4228 FF FF FF FF FF FF FF FF
.:4230 00 00 00 00 00 00 00 00
.:4238 00 00 00 00 00 00 00 00
.:4240 00 18 00 00 3C 00 00 7E
.:4248 00 00 FF 00 01 FF 80 03
.:4250 FF C0 03 FF C0 03 FF C0
.:4258 00 3C 00 00 3C 00 00 3C
.:4260 00 00 3C 00 00 3C 00 00
.:4268 00 00 00 00 00 00 00 00
.:4270 00 00 00 00 00 00 00 00
.:4278 00 00 00 00 00 00 00 00
.:4280 E0 00 00 B0 00 00 98 00
.:4288 00 98 00 00 98 00 00 B0
.:4290 00 00 E0 7C 82 90 40 82
.:4298 98 40 44 88 40 44 88 78
.:42A0 28 88 40 28 88 40 10 88
.:42A8 40 10 88 7C 10 00 00 00
.:42B0 00 00 00 00 00 00 00 00
.:42B8 00 00 00 00 00 00 00 00
.:42C0 00 00 00 00 00 00 00 00
.:42C8 00 00 00 00 00 00 00 00
.:42D0 00 00 7C 88 C3 40 C9 22
.:42D8 40 C9 22 40 A9 02 78 A9
.:42E0 73 40 A9 12 40 99 12 40
.:42E8 99 22 7C 98 C3 00 00 00
.:42F0 00 00 00 00 00 00 00 00
.:42F8 00 00 00 00 00 00 00 00
.:4300 00 00 00 00 00 00 00 00
.:4308 00 00 00 00 00 00 00 00
.:4310 00 00 E0 00 00 00 00 00
.:4318 00 00 00 00 63 C1 C0 92
.:4320 00 00 93 80 00 92 00 E0
.:4328 62 00 00 00 00 00 00 00
.:4330 00 00 00 00 00 00 00 00
.:4338 00 00 00 00 00 00 00 00
.:4340 00 00 07 00 00 05 00 00
.:4348 04 00 00 04 00 00 04 00
.:4350 00 04 00 00 07 00 00 04
.:4358 00 00 04 F2 4F 04 42 48
.:4360 04 43 CE 04 42 48 04 42
.:4368 48 04 42 4F 04 00 00 00
.:4370 00 00 00 00 00 00 00 00
.:4378 00 00 00 00 00 00 00 00
.:4380 00 00 00 80 00 00 40 00
.:4388 00 40 00 00 40 00 00 80
.:4390 00 00 08 8E 3E 08 91 20
.:4398 08 91 20 08 91 20 0F 91
.:43A0 3C 08 91 20 08 91 20 08
.:43A8 91 20 08 8E 3E 00 00 00
.:43B0 00 00 00 00 00 00 00 00
.:43B8 00 00 00 00 00 00 00 00
```

```
.:43C0 00 00 00 00 00 00 00 00
.:43C8 00 00 00 00 00 00 00 00
.:43D0 00 00 44 F9 10 64 21 10
.:43D8 64 21 B0 54 20 A0 54 20
.:43E0 40 54 20 A0 4C 21 B0 4C
.:43E8 21 10 44 F9 10 00 00 00
.:43F0 00 00 00 00 00 00 00 00
.:43F8 00 00 00 00 00 00 00 00
.:4400 80 20 70 80 20 88 80 21
.:4408 04 80 22 02 80 22 02 80
.:4410 22 02 80 22 02 44 42 02
.:4418 44 42 02 2E 82 02 2A 82
.:4420 02 11 01 04 11 00 88 11
.:4428 00 70 00 00 00 00 00 00
.:4430 00 00 00 00 00 00 00 00
.:4438 00 00 00 00 00 00 00 00
.:4440 20 20 00 30 20 00 30 20
.:4448 00 28 20 00 28 20 00 24
.:4450 20 00 22 20 00 22 20 00
.:4458 21 20 00 20 A0 00 20 A0
.:4460 00 20 60 00 20 60 00 20
.:4468 20 00 00 00 00 00 00 00
.:4470 00 00 00 00 00 00 00 00
.:4478 00 00 00 00 00 00 00 00
.:4480 80 23 FC 80 22 00 80 22
.:4488 00 80 22 00 80 22 00 80
.:4490 22 00 80 23 F0 44 42 00
.:4498 44 42 00 2E 82 00 2A 82
.:44A0 00 11 02 00 11 02 00 11
.:44A8 03 FC 00 00 00 00 00 00
.:44B0 00 00 00 00 00 00 00 00
.:44B8 00 00 00 00 00 00 00 00
.:44C0 AA A0 00 AA A0 00 00 00
.:44C8 00 FF F0 00 FF F0 00 00
.:44D0 00 00 55 50 00 55 50 00
.:44D8 00 00 00 AA A0 00 AA A0
.:44E0 00 00 00 00 00 00 00 00
.:44E8 00 00 00 00 00 00 00 00
.:44F0 00 00 00 00 00 00 00 00
.:44F8 00 00 00 00 00 00 00 00
.:4500 AA AA 00 AA AA 00 00 00
.:4508 00 FF F0 00 FF F0 00 00
.:4510 00 00 55 50 00 55 50 00
.:4518 00 00 00 AA A0 00 AA A0
.:4520 00 00 00 00 00 00 00 00
.:4528 00 00 00 00 00 00 00 00
.:4530 00 00 00 00 00 00 00 00
.:4538 00 00 00 00 00 00 00 00
.:4540 AA AA 00 AA AA 00 00 00
.:4548 00 FF FF 00 FF FF 00 00
.:4550 00 00 55 50 00 55 50 00
.:4558 00 00 00 AA A0 00 AA A0
.:4560 00 00 00 00 00 00 00 00
.:4568 00 00 00 00 00 00 00 00
.:4570 00 00 00 00 00 00 00 00
.:4578 00 00 00 00 00 00 00 00
.:4580 AA AA 00 AA AA 00 00 00
.:4588 00 FF FF 00 FF FF 00 00
.:4590 00 00 55 55 00 55 55 00
.:4598 00 00 00 AA A0 00 AA A0
.:45A0 00 00 00 00 00 00 00 00
.:45A8 00 00 00 00 00 00 00 00
.:45B0 00 00 00 00 00 00 00 00
.:45B8 00 00 00 00 00 00 00 00
.:45C0 AA AA 00 AA AA 00 00 00
.:45C8 00 FF FF 00 FF FF 00 00
.:45D0 00 00 55 55 00 55 55 00
.:45D8 00 00 00 AA AA 00 AA AA
.:45E0 00 00 00 00 00 00 00 00
.:45E8 00 00 00 00 00 00 00 00
.:45F0 00 00 00 00 00 00 00 00
.:45F8 00 00 00 00 00 00 00 00
.:4600 AA AA A0 AA AA A0 00 00
.:4608 00 FF FF 00 FF FF 00 00
.:4610 00 00 55 55 00 55 55 00
.:4618 00 00 00 AA AA 00 AA AA
.:4620 00 00 00 00 00 00 00 00
.:4628 00 00 00 00 00 00 00 00
.:4630 00 00 00 00 00 00 00 00
.:4638 00 00 00 00 00 00 00 00
.:4640 AA AA A0 AA AA A0 00 00
.:4648 00 FF FF F0 FF FF F0 00
.:4650 00 00 55 55 00 55 55 00
.:4658 00 00 00 AA AA 00 AA AA
.:4660 00 00 00 00 00 00 00 00
.:4668 00 00 00 00 00 00 00 00
.:4670 00 00 00 00 00 00 00 00
.:4678 00 00 00 00 00 00 00 00
.:4680 AA AA A0 AA AA A0 00 00
.:4688 00 FF FF F0 FF FF F0 00
.:4690 00 00 55 55 50 55 55 50
.:4698 00 00 00 AA AA 00 AA AA
.:46A0 00 00 00 00 00 00 00 00
.:46A8 00 00 00 00 00 00 00 00
.:46B0 00 00 00 00 00 00 00 00
.:46B8 00 00 00 00 00 00 00 00
.:46C0 AA AA A0 AA AA A0 00 00
.:46C8 00 FF FF F0 FF FF F0 00
.:46D0 00 00 55 55 50 55 55 50
.:46D8 00 00 00 AA AA A0 AA AA
.:46E0 A0 00 00 00 00 00 00 00
.:46E8 00 00 00 00 00 00 00 00
.:46F0 00 00 00 00 00 00 00 00
.:46F8 00 00 00 00 00 00 00 00
.:4700 AA AA AA AA AA AA 00 00
.:4708 00 FF FF F0 FF FF F0 00
.:4710 00 00 55 55 50 55 55 50
.:4718 00 00 00 AA AA A0 AA AA
.:4720 A0 00 00 00 00 00 00 00
.:4728 00 00 00 00 00 00 00 00
.:4730 00 00 00 00 00 00 00 00
.:4738 00 00 00 00 00 00 00 00
.:4740 AA AA AA AA AA AA 00 00
.:4748 00 FF FF FF FF FF FF 00
.:4750 00 00 55 55 50 55 55 50
.:4758 00 00 00 AA AA A0 AA AA
.:4760 A0 00 00 00 00 00 00 00
.:4768 00 00 00 00 00 00 00 00
```

```
.:4770 00 00 00 00 00 00 00 00
.:4778 00 00 00 00 00 00 00 00
.:4780 AA AA AA AA AA AA 00 00
.:4788 00 FF FF FF FF FF FF 00
.:4790 00 00 55 55 55 55 55 55
.:4798 00 00 00 AA AA A0 AA AA
.:47A0 A0 00 00 00 00 00 00 00
.:47A8 00 00 00 00 00 00 00 00
.:47B0 00 00 00 00 00 00 00 00
.:47B8 00 00 00 00 00 00 00 00
.:47C0 AA AA AA AA AA AA 00 00
.:47C8 00 FF FF FF FF FF FF 00
.:47D0 00 00 55 55 55 55 55 55
.:47D8 00 00 00 AA AA AA AA AA
.:47E0 AA 00 00 00 00 00 00 00
.:47E8 00 00 00 00 00 00 00 00
.:47F0 00 00 00 00 00 00 00 00
.:47F8 00 00 00 00 00 00 00 00
.:4800 40 15 00 15 55 00 65 A9
.:4808 40 5E AF 50 1B AE 94 6A
.:4810 FA 55 1A EF 50 1E AB 54
.:4818 45 65 54 05 51 40 01 41
.:4820 00 00 00 00 00 00 00 00
.:4828 00 00 00 00 00 00 00 00
.:4830 00 00 00 00 00 00 00 00
.:4838 00 00 00 00 00 00 00 00
.:4840 80 2A 00 2A AA 00 9A FE
.:4848 80 A7 5F A0 2D F7 E8 BF
.:4850 5F AA 2F 75 A0 27 7D A8
.:4858 8A BA A0 0A A2 80 02 82
.:4860 00 00 00 00 00 00 00 00
.:4868 00 00 00 00 00 00 00 00
.:4870 00 00 00 00 00 00 00 00
.:4878 00 00 00 00 00 00 00 00
.:4880 00 00 00 00 00 00 00 00
.:4888 00 00 00 00 00 00 00 00
.:4890 00 00 00 00 00 00 00 00
.:4898 00 00 00 00 00 00 00 00
.:48A0 00 00 00 00 00 00 00 00
.:48A8 00 00 00 00 00 00 00 00
.:48B0 00 00 00 00 00 00 00 00
.:48B8 00 00 00 00 00 00 00 00
.:48C0 00 00 00 00 00 00 00 00
.:48C8 00 00 00 00 00 00 00 00
.:48D0 00 00 00 00 00 00 00 00
.:48D8 00 00 00 00 00 00 00 00
.:48E0 00 00 00 00 00 00 00 00
.:48E8 00 00 00 00 00 00 00 00
.:48F0 00 00 00 00 00 00 00 00
.:48F8 00 00 00 00 00 00 00 00
.:4900 A2 F7 F0 FF 00 FF DF 7F
.:4908 11 FF 00 3F 5D FF 00 F4
.:4910 04 DD D7 00 00 FF 00 00
.:4918 00 FF 00 FF D0 32 00 F3
.:4920 35 A3 22 FF 20 FF F7 FF
.:4928 21 FF 00 F7 00 FF 51 3D
.:4930 B1 F7 D0 FF 00 FF 00 36
.:4938 30 FF 00 FF 00 FD 07 FF
.:4940 00 31 00 FD 00 F7 10 F7
.:4948 31 FF 05 F6 D1 71 31 FF
.:4950 00 F7 D0 FF 00 FF 00 FF
.:4958 00 FF 00 FF 00 AF 10 F5
.:4960 71 FF 00 F5 24 FF 1D FD
.:4968 21 FF 00 FF 00 FF 10 31
.:4970 00 FF 00 FF 00 FF 00 FF
.:4978 00 FF 2F FF 02 FF 01 37
.:4980 D7 FF 00 F5 FD 32 00 FF
.:4988 FF FF 00 FF D7 FF 40 DD
.:4990 11 FF 90 FF 00 FF 00 30
.:4998 11 DD D7 80 00 B7 00 7A
.:49A0 31 F2 00 FD 00 FF 77 FF
.:49A8 71 F3 F0 FF 00 36 00 FF
.:49B0 02 FF 00 FF 00 FF 00 FF
.:49B8 B1 3F 00 F0 80 FF D0 19
.:49C0 00 DD 00 5F 20 F1 00 3D
.:49C8 00 F3 00 FF 02 5D 00 FF
.:49D0 11 DD 00 FD 00 FF 32 FF
.:49D8 50 FF 00 FF 00 FF 00 BF
.:49E0 21 FF 00 F5 D1 31 00 FF
.:49E8 00 FF 00 FF 00 FD 00 F6
.:49F0 31 FF FF F0 00 FF 04 D0
.:49F8 31 73 FF FD F2 33 F6 F0
.:4A00 A3 F7 B0 FF 00 FF DF 7F
.:4A08 11 FF 00 3F 5D FF 00 B4
.:4A10 04 DD 94 00 00 FF 00 00
.:4A18 00 FF 00 FF 90 32 00 F3
.:4A20 35 23 22 FF 20 FF B4 FF
.:4A28 21 FF 00 F7 00 FF 51 3D
.:4A30 31 F7 D0 FF 00 FF 00 32
.:4A38 30 FF 00 FF 00 DD 11 FF
.:4A40 00 31 00 FD 00 F7 10 B4
.:4A48 31 FF 04 B6 D1 71 31 FF
.:4A50 00 F3 D0 FF 00 FF 00 FF
.:4A58 00 FF 00 FF 00 2F 10 F1
.:4A60 B0 FF 00 F5 24 FF 1D FD
.:4A68 21 FF 00 FF 00 FF 10 31
.:4A70 00 FF 00 FF 00 FF 00 FF
.:4A78 00 FF 2F FF 02 FF 00 35
.:4A80 B4 FF 00 F5 BD 32 00 FF
.:4A88 FF FF 00 FF B4 FF 40 DD
.:4A90 11 FF 90 FF 00 FF 00 31
.:4A98 11 DD B4 80 00 B7 00 3A
.:4AA0 31 F2 00 BD 00 FF 77 FF
.:4AA8 71 F3 F0 FF 00 32 10 FF
.:4AB0 23 FF 00 FF 00 FF 00 FF
.:4AB8 B1 3F 00 F0 00 FF 90 19
.:4AC0 00 DD 00 5F 20 F1 00 3D
.:4AC8 00 F3 00 FF 02 5D 00 FF
.:4AD0 11 DD 00 FD 00 FF 32 FF
.:4AD8 50 FF 00 FF 00 FF 00 BF
.:4AE0 21 FF 00 F5 D1 31 00 FF
.:4AE8 00 FF 00 FF 00 FD 00 F2
.:4AF0 31 FF FF B0 00 FF 04 D0
.:4AF8 31 73 FF FD B2 33 B6 B8
.:4B00 A0 F7 30 FF 00 FF DB 7F
.:4B08 11 FF 00 3F 5D FF 00 A4
.:4B10 00 DD 93 00 00 FF 00 00
.:4B18 00 FF 00 FF 80 32 00 F3
.:4B20 31 23 22 FF 20 FB 33 FF
```

```
.:4B28 21 FF 00 F7 00 FF 51 3D
.:4B30 31 F7 D0 FF 00 FF 00 33
.:4B38 30 FF 00 FF 00 DD 03 FF
.:4B40 00 31 00 FD 00 F7 10 93
.:4B48 31 FF 01 A2 D1 71 31 FF
.:4B50 00 F3 D0 FF 00 FF 00 FF
.:4B58 00 FF 00 FF 00 2F 10 F1
.:4B60 31 FF 00 F5 24 FF 0D FD
.:4B68 21 FF 00 FF 00 FF 10 31
.:4B70 00 FF 00 FF 00 FF 00 FF
.:4B78 20 FF 2F FF 02 FF 01 37
.:4B80 13 FF 00 F5 3D 33 00 FF
.:4B88 FF FF 00 FF 12 FF 40 DD
.:4B90 11 FF 80 FF 00 FF 00 31
.:4B98 11 DD 13 80 00 B7 00 72
.:4BA0 31 F2 00 7D 00 FF 77 FF
.:4BA8 71 F3 F0 FF 00 33 00 F7
.:4BB0 00 FF 00 FF 00 FF 00 FF
.:4BB8 B1 3F 00 F0 00 FF 10 19
.:4BC0 00 DD 00 5F 20 F1 00 3D
.:4BC8 00 F3 00 FF 02 59 00 FF
.:4BD0 11 DD 00 FD 00 FF 32 FF
.:4BD8 40 FF 00 FF 00 FF 00 2F
.:4BE0 21 FF 00 F5 D1 31 00 FF
.:4BE8 00 FF 00 FF 00 FD 00 F2
.:4BF0 31 FF FF B0 00 FF 00 D0
.:4BF8 31 73 FF FD B2 33 A2 90
.:4C00 AA 00 00 AA 00 00 AA 00
.:4C08 00 00 00 00 FF 00 00 FF
.:4C10 00 00 FF 00 00 00 00 00
.:4C18 55 00 00 55 00 00 55 00
.:4C20 00 00 00 00 AA 00 00 AA
.:4C28 00 00 AA 00 00 00 00 00
.:4C30 00 00 00 00 00 00 00 00
.:4C38 00 00 00 00 00 00 00 00
.:4C40 00 31 00 FD 00 F7 10 F7
.:4C48 31 FF 05 E6 D1 71 31 FF
.:4C50 00 F3 D0 FF 00 FF 00 FF
.:4C58 00 FF 00 FF 00 2F 10 F1
.:4C60 F1 FF 00 F5 24 FF 0D FD
.:4C68 21 FF 00 FF 00 FF 10 31
.:4C70 00 FF 00 FF 00 FF 00 FF
.:4C78 00 FF 2F FF 02 FF 01 37
.:4C80 F7 FF 00 F5 FD 32 00 FF
.:4C88 FF FF 00 FF D7 FF 40 DD
.:4C90 11 FF 80 FF 00 FF 00 31
.:4C98 11 DD D7 80 00 B7 00 7A
.:4CA0 31 F2 00 FD 00 FF 77 FF
.:4CA8 71 F3 F0 FF 00 32 00 FF
.:4CB0 23 FF 00 FF 00 FF 00 FF
.:4CB8 B1 3F 00 F0 00 FF D0 19
.:4CC0 00 DD 00 5F 20 F1 00 3D
.:4CC8 00 F3 00 FF 02 5D 00 FF
.:4CD0 11 DD 00 FD 00 FF 32 FF
.:4CD8 40 FF 00 FF 00 FF 00 BF
.:4CE0 21 FF 00 F5 D1 31 00 FF
.:4CE8 00 FF 00 FF 00 FD 00 F2
.:4CF0 31 FF FF F0 00 FF 00 D0
.:4CF8 31 73 FF FD F2 33 E6 D0
```

```
.:4D00 A2 F7 70 FF 00 FF DB 7F
.:4D08 01 FF 00 3F 5D FF 00 E6
.:4D10 00 DD 53 00 00 FF 00 00
.:4D18 00 FF 00 FF C0 22 00 F3
.:4D20 21 23 22 FF 20 FB 73 FF
.:4D28 21 FF 00 F7 00 FF 51 3D
.:4D30 21 F7 D0 FF 00 FF 00 22
.:4D38 30 FF 00 FF 00 DD 03 FF
.:4D40 00 21 00 FD 00 F7 00 53
.:4D48 21 FF 01 E2 D1 71 21 FF
.:4D50 00 F3 D0 FF 00 FF 00 FF
.:4D58 00 FF 00 FF 00 2F 10 F1
.:4D60 71 FF 00 F5 24 FF 0D FD
.:4D68 21 FF 00 FF 00 FF 10 21
.:4D70 00 FF 00 FF 00 FF 00 FF
.:4D78 00 FF 2F FF 02 FF 01 27
.:4D80 53 FF 00 F5 7D 23 00 FF
.:4D88 FF FF 00 FF 53 FF 40 DD
.:4D90 01 FF 90 FF 00 FF 00 21
.:4D98 01 DD 53 80 00 A7 00 72
.:4DA0 21 F2 00 7D 00 FF 77 FF
.:4DA8 71 F3 F1 FF 00 23 00 F7
.:4DB0 02 FF 00 FF 00 FF 00 FF
.:4DB8 21 3F 00 F0 00 FF 50 09
.:4DC0 00 DF 00 5F 20 F1 00 2D
.:4DC8 00 F3 00 FF 02 59 00 FF
.:4DD0 01 DF 00 FD 00 FF 12 FF
.:4DD8 50 FF 00 FF 00 FF 00 2F
.:4DE0 21 FF 00 F5 D1 21 00 FF
.:4DE8 00 FF 00 FF 00 FD 00 F2
.:4DF0 21 FF FF F2 00 FF 00 D0
.:4DF8 31 73 FF FD F2 23 E2 D0
.:4E00 A0 F7 70 FF 00 FF DF 7F
.:4E08 11 FF 00 3F 5D FF 00 F4
.:4E10 04 DD D7 00 00 FF 00 00
.:4E18 00 FF 00 FF D0 32 00 F3
.:4E20 35 23 22 FF 20 FF 77 FF
.:4E28 21 FF 00 F7 00 FF 51 3D
.:4E30 31 F7 D0 FF 00 FF 00 3E
.:4E38 30 FF 00 FF 00 DD 15 FF
.:4E40 00 31 00 FD 00 F7 10 D7
.:4E48 31 FF 05 F6 D1 71 31 FF
.:4E50 00 F3 D0 FF 00 FF 00 FF
.:4E58 00 FF 00 FF 00 2F 10 F5
.:4E60 71 FF 00 F5 24 FF 1D FD
.:4E68 21 FF 00 FF 00 FF 10 31
.:4E70 00 FF 00 FF 00 FF 00 FF
.:4E78 00 FF 2F FF 02 FF 01 37
.:4E80 57 FF 00 F5 FD 32 00 FF
.:4E88 FF FF 00 FF 56 FF 40 DD
.:4E90 11 FF 90 FF 00 FF 00 31
.:4E98 11 DD 57 80 00 B7 00 7A
.:4EA0 31 F2 00 FD 00 FF 77 FF
.:4EA8 71 F3 F0 FF 00 36 00 FF
.:4EB0 20 FF 00 FF 00 FF 00 FF
.:4EB8 B1 3F 00 F0 00 FF 50 19
.:4EC0 00 DD 00 5F 00 F1 00 3D
.:4EC8 00 F3 00 FF 02 5D 00 FF
.:4ED0 11 DD 00 FD 00 FF 12 FF
```

```
.:4ED8 50 FF 00 FF 00 FF 00 BF
.:4EE0 21 FF 00 F5 D1 31 00 FF
.:4EE8 00 FF 00 FF 00 FD 00 F2
.:4EF0 31 FF FF F0 00 FF 04 D0
.:4EF8 31 73 FF FD F2 33 F6 DC
.:4F00 AB F7 F8 FF 00 FF DB FF
.:4F08 81 FF 00 BF 5D FF 00 A6
.:4F10 00 DD DF 00 00 FF 00 00
.:4F18 00 FF 00 FF 80 A2 00 FB
.:4F20 A5 23 22 FF 20 FF F3 FF
.:4F28 21 FF 00 F7 00 FF 51 3D
.:4F30 A1 F7 D0 FF 00 FF 00 AF
.:4F38 30 FF 00 FF 00 FD 81 FF
.:4F40 00 A1 00 FD 00 F7 00 FF
.:4F48 A1 FF 0D A6 D1 F1 21 FF
.:4F50 00 F3 D0 FF 00 FF 00 FF
.:4F58 00 FF 00 FF 00 2F 90 F1
.:4F60 71 FF 00 F5 24 FF 8D FD
.:4F68 21 FF 00 FF 00 FF 90 A1
.:4F70 00 FF 00 FF 00 FF 00 FF
.:4F78 00 FF 2F FF 02 FF 01 A7
.:4F80 7F FF 00 F5 FD AF 00 FF
.:4F88 FF FF 00 FF D7 FF 40 DD
.:4F90 81 FF 80 FF 00 FF 00 A3
.:4F98 81 DD FF 80 00 AF 00 FA
.:4FA0 A1 F2 00 FD 00 FF 77 FF
.:4FA8 71 F3 F1 FF 00 AF 00 FF
.:4FB0 0B FF 81 FF 00 FF 00 FF
.:4FB8 A1 3F 00 F0 00 FF D0 89
.:4FC0 00 DD 00 DF 20 F1 00 AF
.:4FC8 00 F3 00 FF 02 DD 00 FF
.:4FD0 81 DD 00 FD 00 FF 92 FF
.:4FD8 50 FF 00 FF 00 FF 00 AF
.:4FE0 21 FF 00 F5 D1 A1 00 FF
.:4FE8 00 FF 00 FF 00 FD 00 F2
.:4FF0 A1 FF FF F2 00 FF 00 D0
.:4FF8 B1 F3 FF FD F2 23 A6 F9
.:5000 A2 F7 E0 FF 00 FF DB FF
.:5008 80 FF 00 BF 5D FF 00 E4
.:5010 00 DD C3 00 00 FF 00 00
.:5018 00 FF 00 FF C0 A2 00 F2
.:5020 A0 22 22 FF 20 FB E3 FF
.:5028 20 FF 00 F7 00 FF 40 3D
.:5030 A0 F7 D0 FF 00 FF 00 A3
.:5038 30 FF 00 FF 00 DD 01 FF
.:5040 00 A0 00 ED 00 F6 00 C3
.:5048 A0 FF 01 E2 D1 F0 20 FF
.:5050 00 F3 D0 FF 00 FF 00 FF
.:5058 00 FF 00 FF 00 2F 80 F1
.:5060 60 FF 00 F5 24 FF 01 FD
.:5068 20 FF 00 FF 00 FF 90 A0
.:5070 00 FF 00 FF 00 FF 00 FF
.:5078 00 FF 2F FF 02 FF 01 A6
.:5080 C3 FF 00 F5 ED A2 00 FF
.:5088 FF FF 00 FF 83 FF 40 DD
.:5090 80 FF 80 FF 00 FF 00 A0
.:5098 80 DD C3 80 00 A7 00 F2
.:50A0 A0 F2 00 FD 00 FF 77 FF
.:50A8 71 F3 F0 FF 00 A7 00 F7
.:50B0 82 FF 80 FF 00 FF 00 FF
.:50B8 A0 3F 00 E0 00 FF 80 88
.:50C0 00 DD 00 DF 20 F0 00 AC
.:50C8 00 F2 00 FF 02 D9 00 FF
.:50D0 80 DD 00 FD 00 FF 92 FF
.:50D8 40 FF 00 FF 00 FF 00 AF
.:50E0 20 FF 00 F5 D1 A0 00 FF
.:50E8 00 FF 00 FF 00 FD 00 E2
.:50F0 A0 FF FF F0 00 FF 00 D0
.:50F8 A0 F3 FF FD F2 22 E2 D0
.:5100 A1 F7 30 FF 00 FF DF 7F
.:5108 11 FF 00 3F 5D FF 00 B4
.:5110 04 DD 93 00 00 FF 00 00
.:5118 00 FF 00 FF 90 32 00 F3
.:5120 35 23 22 FF 20 FF 33 FF
.:5128 21 FF 00 F7 00 FF 51 3D
.:5130 31 F7 D0 FF 00 FF 00 3B
.:5138 30 FF 00 FF 00 DD 11 FF
.:5140 00 31 00 FD 00 F7 10 BB
.:5148 31 FF 0D B2 D1 71 31 FF
.:5150 00 F3 D0 FF 00 FF 00 FF
.:5158 00 FF 00 FF 00 2F 10 F1
.:5160 B1 FF 00 F5 24 FF 1D FD
.:5168 21 FF 00 FF 00 FF 10 31
.:5170 00 FF 00 FF 00 FF 00 FF
.:5178 00 FF 2F FF 02 FF 01 35
.:5180 BF FF 10 F5 BD 30 00 FF
.:5188 FF FF 00 FF 37 FF 40 DD
.:5190 11 FF 90 FF 00 FF 00 31
.:5198 11 DD B3 80 00 B7 00 7A
.:51A0 31 F2 00 FD 00 FF 77 FF
.:51A8 71 F3 F0 FF 00 33 10 FF
.:51B0 21 FF 00 FF 00 FF 00 FF
.:51B8 B1 3F 00 F0 00 FF 90 19
.:51C0 00 DD 00 5F 20 F1 00 3D
.:51C8 00 F3 00 FF 02 5D 00 FF
.:51D0 11 DD 00 FD 00 FF 32 FF
.:51D8 50 FF 00 FF 00 FF 00 BF
.:51E0 21 FF 00 F5 D1 31 00 FF
.:51E8 00 FF 00 FF 00 FD 00 F2
.:51F0 31 FF FF F0 00 FF 04 D0
.:51F8 31 73 FF FD F2 33 B2 B9
.:5200 A2 F7 B0 FF 00 FF DB 7F
.:5208 10 FF 00 3F 5D FF 00 A4
.:5210 00 DD 93 00 00 FF 00 00
.:5218 00 FF 00 FF 80 32 00 F2
.:5220 30 22 22 FF 20 FB B3 FF
.:5228 20 FF 00 F7 00 FF 50 3D
.:5230 B0 F7 D0 FF 00 FF 00 3B
.:5238 30 FF 00 FF 00 FD 03 FF
.:5240 00 30 00 FD 00 F6 10 B3
.:5248 30 FF 01 A2 D1 70 30 FF
.:5250 00 F3 D0 FF 00 FF 00 FF
.:5258 00 FF 00 FF 00 2F 10 F1
.:5260 B0 FF 00 F5 24 FF 0D FD
.:5268 20 FF 00 FF 00 FF 10 30
.:5270 00 FF 00 FF 00 FF 00 FF
.:5278 00 FF 2F FF 02 FF 01 34
.:5280 B3 FF 00 F5 BD 32 00 FF
```

```
.:5288 FF FF 00 FF B3 FF 40 DD
.:5290 10 FF 90 FF 00 FF 00 31
.:5298 10 DD B3 80 00 B7 00 3A
.:52A0 30 F2 00 BD 00 FF 77 FF
.:52A8 71 F3 F0 FF 00 33 00 FF
.:52B0 22 FF 00 FF 00 FF 00 FF
.:52B8 B0 3F 00 F0 00 FF 90 18
.:52C0 00 DD 00 5F 20 F0 00 3D
.:52C8 00 F2 00 FF 02 59 00 FF
.:52D0 11 DD 00 FD 00 FF 32 FF
.:52D8 40 FF 00 FF 00 FF 00 BF
.:52E0 20 FF 00 F5 D1 30 00 FF
.:52E8 00 FF 00 FF 00 FD 00 F2
.:52F0 30 FF FF B0 00 FF 00 D0
.:52F8 30 73 FF FD B2 32 A2 B9
.:5300 A2 F7 F0 FF 00 FF DF 7F
.:5308 11 FF 00 3F 5D FF 00 F4
.:5310 04 DD D7 00 00 FF 00 00
.:5318 00 FF 00 FF D0 32 00 F3
.:5320 35 A3 22 FF 20 FF F7 FF
.:5328 21 FF 00 F7 00 FF D1 3D
.:5330 B1 F7 D0 FF 00 FF 00 33
.:5338 30 FF 00 FF 00 DD 53 FF
.:5340 00 31 00 FD 00 F7 10 F7
.:5348 31 FF 05 F6 D1 71 31 FF
.:5350 00 F3 D0 FF 00 FF 00 FF
.:5358 00 FF 00 FF 00 AF 10 F1
.:5360 F1 FF 00 F5 24 FF 5D FD
.:5368 21 FF 00 FF 00 FF 10 31
.:5370 00 FF 00 FF 00 FF 00 FF
.:5378 00 FF 2F FF 02 FF 01 37
.:5380 F7 FF 00 F5 FD 32 00 FF
.:5388 FF FF 00 FF F7 FF 40 DD
.:5390 11 FF 90 FF 00 FF 00 31
.:5398 11 DD F7 80 00 B7 00 7A
.:53A0 31 F2 00 FD 00 FF 77 FF
.:53A8 71 F3 F0 FF 00 32 00 F7
.:53B0 22 FF 00 FF 00 FF 00 FF
.:53B8 B1 3F 00 F0 80 FF D0 19
.:53C0 00 DD 00 5F 20 F1 00 3D
.:53C8 00 F3 00 FF 02 5D 00 FF
.:53D0 11 DD 00 FD 00 FF 32 FF
.:53D8 50 FF 00 FF 00 FF 00 BF
.:53E0 21 FF 00 F5 D1 31 00 FF
.:53E8 00 FF 00 FF 00 FD 00 F2
.:53F0 31 FF FF F0 00 FF 04 D0
.:53F8 31 73 FF FD F2 33 F6 F0
.:5400 F2 F7 B0 FF 00 FF DF 7F
.:5408 10 FF 00 3F 5D FF 00 F4
.:5410 00 DD 97 00 00 FF 00 00
.:5418 00 FF 00 FF D0 32 00 F2
.:5420 34 A2 22 FF 20 FF B3 FF
.:5428 20 FF 00 F7 00 FF D0 3D
.:5430 B0 F7 D0 FF 00 FF 00 32
.:5438 30 FF 00 FF 00 DD 13 FF
.:5440 00 30 00 FD 00 F6 10 B7
.:5448 30 FF 05 F6 D1 70 30 FF
.:5450 00 F3 D0 FF 00 FF 00 FF
.:5458 00 FF 00 FF 00 AF 10 F1
.:5460 F0 FF 00 F5 24 FF 1D FD
.:5468 20 FF 00 FF 00 FF 10 30
.:5470 00 FF 00 FF 00 FF 00 FF
.:5478 00 FF 2F FF 02 FF 01 34
.:5480 B7 FF 00 F5 BD 32 00 FF
.:5488 FF FF 00 FF B7 FF 40 DD
.:5490 10 FF 90 FF 00 FF 00 30
.:5498 10 DD B7 80 00 B7 00 7A
.:54A0 30 F2 00 FD 00 FF 77 FF
.:54A8 71 F3 F0 FF 00 32 00 FF
.:54B0 72 FF 00 FF 00 FF 00 FF
.:54B8 B0 3F 00 F0 80 FF 90 18
.:54C0 00 DD 00 5F 20 F0 00 3C
.:54C8 00 F2 00 FF 02 5D 00 FF
.:54D0 10 DD 00 FD 00 FF 32 FF
.:54D8 50 FF 00 FF 00 FF 00 BF
.:54E0 20 FF 00 F5 D1 30 00 FF
.:54E8 00 FF 00 FF 00 FD 00 F2
.:54F0 30 FF FF F0 00 FF 04 D0
.:54F8 30 73 FF FD F2 32 F6 B8
.:5500 A0 F7 F8 FF 00 FF DF 7F
.:5508 10 FF 00 3F 5D FF 00 E4
.:5510 00 DD DD 00 00 FF 00 00
.:5518 00 FF 00 FF C0 32 00 FA
.:5520 34 23 22 FF 20 FF F7 FF
.:5528 20 FF 00 F7 00 FF 51 3D
.:5530 30 F7 D0 FF 00 FF 00 3A
.:5538 30 FF 00 FF 00 DD 09 FF
.:5540 00 30 00 FD 00 F6 10 DF
.:5548 30 FF 0D E6 D1 70 30 FF
.:5550 00 F3 D0 FF 00 FF 00 FF
.:5558 00 FF 00 FF 00 2F 10 F1
.:5560 70 FF 00 F5 24 FF 4D FD
.:5568 20 FF 00 FF 00 FF 10 30
.:5570 00 FF 00 FF 00 FF 00 FF
.:5578 00 FF 2F FF 00 FF 01 34
.:5580 DD FF 00 F5 FD 38 00 FF
.:5588 FF FF 00 FF D5 FF 40 DD
.:5590 10 FF 80 FF 00 FF 00 30
.:5598 10 DD DD 80 00 BF 00 7A
.:55A0 30 F2 00 FD 00 FF 77 FF
.:55A8 71 F3 F0 FF 00 38 00 FF
.:55B0 20 FF 00 FF 00 FF 00 FF
.:55B8 B0 3F 00 F0 00 FF D0 18
.:55C0 00 DD 00 5F 20 F0 00 3C
.:55C8 00 F2 00 FF 02 59 00 FF
.:55D0 10 DD 00 FD 00 FF 32 FF
.:55D8 40 FF 00 FF 00 FF 00 BF
.:55E0 20 FF 00 F5 D1 30 00 FF
.:55E8 00 FF 00 FF 00 FD 00 F2
.:55F0 30 FF FF F0 00 FF 00 D0
.:55F8 30 73 FF FD F2 32 E6 D8
.:5600 EF FF E7 22 EF 20 FF FB
.:5608 EF 71 FF 00 FF 15 FF 20
.:5610 EF 30 FF 7F FF 20 FF 11
.:5618 FF 38 FF 04 FF 73 7C FF
.:5620 CF 32 FF 7B FF 00 FF 00
.:5628 FF 02 FF 10 7B 30 FF FF
.:5630 FF FB FF 30 FF 00 FF 31
```

```
.:5638 7B 20 14 20 73 31 FF 73
.:5640 FF 38 EF 30 FF 00 FF FF
.:5648 EF 10 FF D3 FF 00 DF FB
.:5650 FF 00 DF 00 FF 00 11 30
.:5658 FF 00 FB FF FD 0D FD 3D
.:5660 FD 0D FD 10 FD FD FD 3D
.:5668 FD FD FF FB 7F 00 FF FF
.:5670 FF 00 10 02 FF 00 CF 20
.:5678 FF 00 4C 10 F3 30 57 00
.:5680 0D 6D D3 D3 0D FD 0D ED
.:5688 1D FF 08 FF 20 FD 0D DF
.:5690 0D FD D3 D3 0D FD 00 FF
.:5698 28 FF 00 FB 00 CF DB FF
.:56A0 38 FF 00 FF 00 FD 0D D3
.:56A8 0D FD 0D EF FF FF 01 FC
.:56B0 20 FF 20 FF 10 FF 00 FF
.:56B8 FF FF FF DF ED 1D D3 ED
.:56C0 0D E7 30 EF 00 FF 10 EF
.:56C8 20 FF 30 FD 0D D3 2D CD
.:56D0 20 FF FB FF 00 FF FF FF
.:56D8 30 FF B4 30 00 FD CF FF
.:56E0 00 FF 00 FF 00 04 2D FD
.:56E8 D3 0D 0D FF 80 FF 03 88
.:56F0 38 81 FD D3 0D FF B2 FF
.:56F8 00 FF FB FF 4C FF FB FF
.:5700 FF FF F7 22 FF 20 FF F7
.:5708 FF 71 FF 00 FF 14 FF 20
.:5710 FF 30 D3 7D FF 20 FF 51
.:5718 F1 18 3D 14 FF 73 FC FF
.:5720 CF 32 FF 73 FF 00 FF 00
.:5728 FF 02 FF 10 7B 30 F7 FF
.:5730 FF F3 FF 30 FF 00 FF 30
.:5738 7B 20 14 39 9D 18 18 78
.:5740 39 38 39 38 F8 08 18 FF
.:5748 FF 10 FF D3 FF 00 FF F3
.:5750 FF 00 DF 00 FF 00 51 30
.:5758 FF 00 FB FF FF 00 FB 30
.:5760 F7 07 78 78 78 97 98 97
.:5768 F3 F9 F9 F9 79 09 F9 FF
.:5770 FF 00 14 00 FF 00 CF 20
.:5778 FF 00 44 10 F3 30 57 00
.:5780 40 65 49 FD 00 FF 00 C7
.:5788 09 89 89 F8 38 39 39 19
.:5790 09 FF 39 C9 09 FF 08 FF
.:5798 30 FF 00 FB 00 FF DB FF
.:57A0 30 FF 00 FF 00 FF 00 FF
.:57A8 00 FF 00 FF FF FF 08 F9
.:57B0 87 89 89 87 78 79 79 97
.:57B8 F9 F9 F9 D9 F9 F8 09 E9
.:57C0 48 E8 30 EF 00 FF 10 EF
.:57C8 20 FF 30 FF 00 FF 22 78
.:57D0 87 87 F8 FF 08 F8 F9 F9
.:57D8 39 F8 39 39 79 78 87 89
.:57E0 08 F9 09 F8 08 48 28 F9
.:57E8 28 09 00 FF 80 FF 03 48
.:57F0 30 FF FF FF 00 FF 38 F8
.:57F8 89 79 78 78 48 F9 F9 78
.:5800 79 79 79 79 87 89 F8 F9
.:5808 E8 F9 F9 09 F9 19 F9 29
.:5810 E9 39 F8 7F FF 20 FF 11
.:5818 FD 30 FF 34 FF 78 38 F9
.:5820 C9 39 FD 79 78 79 87 89
.:5828 89 08 F8 18 78 38 F9 F8
.:5830 F8 F9 F9 39 F9 09 F9 39
.:5838 78 39 19 38 73 31 FF 73
.:5840 FD 38 EF 30 F8 08 F9 F8
.:5848 78 79 79 79 79 89 87 87
.:5850 F7 07 78 08 F8 08 18 39
.:5858 FF 08 F9 F9 F9 09 F9 39
.:5860 F9 09 FF 18 F8 FF FF 30
.:5868 FF F8 F8 87 78 09 F9 78
.:5870 89 78 19 09 F9 09 C9 29
.:5878 F8 87 89 89 F9 78 58 08
.:5880 50 78 59 FD 19 F9 09 E9
.:5888 19 F9 08 F9 38 FF 02 DF
.:5890 18 F8 38 79 89 78 08 F8
.:5898 29 F9 08 F8 09 E9 79 F7
.:58A0 37 F7 09 89 89 F8 09 FF
.:58A8 00 F8 09 F9 F9 F8 09 F9
.:58B0 29 F9 29 F9 19 F8 00 F8
.:58B8 F8 F9 79 89 79 79 03 E9
.:58C0 48 E8 30 E8 09 F9 18 87
.:58C8 89 89 87 F7 78 F8 28 C8
.:58D0 20 F8 F9 F9 09 F9 F9 F9
.:58D8 30 F9 F9 53 39 F8 D8 F8
.:58E0 08 F9 09 F9 79 89 29 F9
.:58E8 28 00 00 F7 87 F7 07 07
.:58F0 37 87 89 89 08 F8 F8 F8
.:58F8 0C FC C8 C9 C9 F9 F8 F9
.:5900 FD F9 F9 39 39 53 FD FF
.:5908 F9 79 F9 00 F9 79 78 28
.:5910 F9 30 FF 78 89 87 87 87
.:5918 78 78 78 79 F9 79 78 FF
.:5920 CF 9C F9 9C F9 39 93 09
.:5928 F9 09 FA FA FA 39 F9 F9
.:5930 F9 F8 F9 39 F8 08 F9 39
.:5938 7B 20 83 29 79 39 79 79
.:5940 78 78 F7 39 89 09 F9 F9
.:5948 FC 1C F9 DC FA 0F FA FF
.:5950 FF 0F CF C9 FC 00 58 39
.:5958 F9 00 F8 F8 F8 09 F8 38
.:5960 FF 00 35 37 F7 F7 F7 39
.:5968 FF FF FF FC CF FC FC FC
.:5970 FC 0C CF 0F FF 0C CF 9F
.:5978 F9 0C 4C 5F F5 CF FC CF
.:5980 FC CF C9 FC 80 FF 07 C7
.:5988 87 FC 7F F9 9F FC CF FC
.:5990 CF FC 5F C5 9C F9 9F FF
.:5998 FC FF CF FB 9F CF DB F7
.:59A0 B7 7C 7C F5 9F F9 0C FF
.:59A8 0C FF 0C FC F7 F7 09 F9
.:59B0 7F FC C9 FC C9 FC C9 FC
.:59B8 F9 FC FC DF F9 DF CF E8
.:59C0 FC EC 3C EF 9F F7 17 E7
.:59C8 A7 F7 FC FF F9 FF 2F CF
.:59D0 2F FF FB FF 0F FF FF FF
.:59D8 3C FF BC 3C 0F FF DF FF
.:59E0 0C FF 9F F9 FC 4C 0F F8
```

```
.:59E8 FF FF 00 FF 00 FD 00 F2
.:59F0 31 FF FF F0 00 FF 00 D0
.:59F8 31 73 FF FD F2 33 B2 98
.:5A00 A2 F7 70 FF 00 FF DB 7F
.:5A08 11 FF 00 3F 5D FF 00 E4
.:5A10 00 DD 53 00 00 FF 00 00
.:5A18 00 FF 00 FF C0 32 00 F3
.:5A20 31 23 22 FF 20 FB 73 FF
.:5A28 21 FF 00 F7 00 FF 51 3D
.:5A30 31 F7 D0 FF 00 FF 00 33
.:5A38 30 FF 00 FF 00 DD 13 FF
.:5A40 00 31 00 FD 00 F7 10 53
.:5A48 31 FF 01 E2 D1 71 31 FF
.:5A50 00 F3 D0 7F 00 FF 00 FF
.:5A58 00 FF 00 FF 00 2F 10 F1
.:5A60 71 FF 00 F5 24 FF 1D FD
.:5A68 21 FF 00 FF 00 FF 10 31
.:5A70 00 FF 00 FF 00 FF 00 FF
.:5A78 00 FF 2F FF 02 FF 01 37
.:5A80 53 FF 00 F5 FD 32 00 FF
.:5A88 FF FF 00 FF 53 FF 40 DD
.:5A90 11 FF 90 FF 00 FF 00 31
.:5A98 11 DD 53 80 00 B7 00 72
.:5AA0 31 F2 00 FD 00 FF 77 FF
.:5AA8 71 F3 F0 FF 00 32 00 F7
.:5AB0 22 FF 00 FF 00 FF 00 FF
.:5AB8 31 3F 00 F0 00 FF 50 19
.:5AC0 00 DD 00 5F 20 F1 00 3D
.:5AC8 00 F3 00 FF 02 59 00 FF
.:5AD0 11 DD 00 FD 00 FF 32 FF
.:5AD8 50 FF 00 FF 00 FF 00 3F
.:5AE0 21 FF 00 F5 D1 31 00 FF
.:5AE8 00 FF 00 FF 00 FD 00 F2
.:5AF0 31 FF FF B0 00 FF 00 D0
.:5AF8 31 73 FF FD B2 33 E2 D1
.:5B00 A2 F7 70 FF 00 FF DB 7F
.:5B08 01 FF 00 3F 5D FF 00 E4
.:5B10 00 DD 57 00 00 FF 00 00
.:5B18 00 FF 00 FF C0 22 00 F3
.:5B20 21 23 22 FF 20 FB 73 FF
.:5B28 21 FF 00 F7 00 FF 51 3D
.:5B30 21 F7 D0 FF 00 FF 00 2A
.:5B38 30 FF 00 FF 00 DD 03 FF
.:5B40 00 21 00 FD 00 F7 00 7F
.:5B48 21 FF 0D E6 D1 71 21 FF
.:5B50 00 F3 D0 FF 00 FF 00 FF
.:5B58 00 FF 00 FF 00 2F 10 F1
.:5B60 71 FF 00 F5 24 FF 0D FD
.:5B68 21 FF 00 FF 00 FF 10 21
.:5B70 00 FF 00 FF 00 FF 00 FF
.:5B78 00 FF 2F FF 02 FF 01 27
.:5B80 7F FF 00 F5 FD 23 00 FF
.:5B88 FF FF 00 FF 57 FF 40 DD
.:5B90 01 FF 80 FF 00 FF 00 21
.:5B98 01 DD 4B 80 00 B7 00 7A
.:5BA0 21 F2 00 FD 00 FF 77 FF
.:5BA8 71 F3 F1 FF 00 23 00 FF
.:5BB0 02 FF 00 FF 00 FF 00 FF
.:5BB8 21 3F 00 F0 00 FF 50 09
.:5BC0 00 DF 00 5F 20 F1 00 2D
.:5BC8 00 F3 00 FF 02 59 00 FF
.:5BD0 01 DF 00 FD 00 FF 32 FF
.:5BD8 40 FF 00 FF 00 FF 00 AF
.:5BE0 21 FF 00 F5 D1 21 00 FF
.:5BE8 00 FF 00 FF 00 FD 00 F2
.:5BF0 21 FF FF F0 00 FF 00 D0
.:5BF8 31 73 FF FD F2 23 E6 FA
.:5C00 00 00 07 00 00 00 00 00
.:5C08 00 71 00 00 00 10 00 00
.:5C10 00 30 00 70 00 00 00 11
.:5C18 00 38 00 00 00 73 70 00
.:5C20 00 30 00 70 00 00 00 00
.:5C28 00 00 00 10 70 30 00 00
.:5C30 00 00 00 30 00 00 00 31
.:5C38 70 00 10 00 73 31 00 73
.:5C40 00 38 00 30 00 00 00 00
.:5C48 00 10 00 D3 00 00 D0 00
.:5C50 00 00 D0 00 00 00 11 30
.:5C58 00 00 00 00 0D 0D 0D 3D
.:5C60 0D 0D 0D 10 0D 0D 0D 3D
.:5C68 0D 0D 00 00 70 00 00 00
.:5C70 00 00 10 00 00 00 00 00
.:5C78 00 00 00 10 03 30 07 00
.:5C80 0D 0D D3 D3 0D 0D 0D 0D
.:5C88 1D 00 08 00 00 0D 0D D0
.:5C90 0D 0D D3 D3 0D 0D 00 00
.:5C98 08 00 00 00 00 00 D0 00
.:5CA0 38 00 00 00 00 0D 0D D3
.:5CA8 0D 0D 0D 00 00 00 01 00
.:5CB0 00 00 00 00 10 00 00 00
.:5CB8 00 00 00 D0 0D 1D D3 0D
.:5CC0 0D 07 30 00 00 00 10 00
.:5CC8 00 00 30 0D 0D D3 0D 0D
.:5CD0 00 00 00 00 00 00 20 00
.:5CD8 30 00 00 30 00 0D 00 00
.:5CE0 00 00 00 00 00 00 0D 0D
.:5CE8 D3 0D 0D 00 80 00 03 88
.:5CF0 38 81 0D D3 0D 00 20 00
.:5CF8 00 00 00 00 00 00 00 00
.:5D00 00 00 07 00 00 00 00 07
.:5D08 00 71 00 00 00 10 00 00
.:5D10 00 30 D3 7D 00 00 00 01
.:5D18 01 18 3D 14 FF 73 FC FF
.:5D20 CF 32 FF 73 FF 00 FF 00
.:5D28 FF 02 FF 10 7B 30 F7 FF
.:5D30 FF F3 FF 30 FF 00 FF 30
.:5D38 7B 20 14 39 9D 18 18 78
.:5D40 39 38 39 38 F8 08 18 FF
.:5D48 FF 10 FF D3 FF 00 FF F3
.:5D50 FF 00 DF 00 FF 00 51 30
.:5D58 FF 00 FB FF FF 00 FB 30
.:5D60 F7 07 78 78 78 97 98 97
.:5D68 F3 F9 F9 F9 79 09 F9 FF
.:5D70 FF 00 14 00 FF 00 CF 20
.:5D78 FF 00 44 10 F3 30 57 00
.:5D80 40 65 49 FD 00 FF 00 C7
.:5D88 09 89 89 F8 38 39 39 19
.:5D90 09 FF 39 C9 09 FF 08 FF
```

```
.:5D98 30 FF 00 FB 00 FF DB FF
.:5DA0 30 FF 00 FF 00 FF 00 FF
.:5DA8 00 FF 00 FF FF FF 08 F9
.:5DB0 87 89 89 87 78 79 79 97
.:5DB8 F9 F9 F9 D9 F9 F8 09 E9
.:5DC0 48 E8 30 EF 00 FF 10 EF
.:5DC8 20 FF 30 FF 00 FF 22 78
.:5DD0 87 87 F8 FF 08 F8 F9 F9
.:5DD8 39 F8 39 39 79 78 87 89
.:5DE0 08 F9 09 F8 08 48 28 F9
.:5DE8 28 09 00 FF 80 FF 03 48
.:5DF0 30 FF FF FF 00 FF 38 F8
.:5DF8 89 79 78 78 48 F9 F9 78
.:5E00 79 79 79 79 87 89 F8 F9
.:5E08 E8 F9 F9 09 F9 19 F9 29
.:5E10 E9 39 F8 7F FF 20 FF 11
.:5E18 FD 30 FF 34 FF 78 38 F9
.:5E20 C9 39 FD 79 78 79 87 89
.:5E28 89 08 F8 18 78 38 F9 F8
.:5E30 F8 F9 F9 39 F9 09 F9 39
.:5E38 78 39 19 38 73 31 FF 73
.:5E40 FD 38 EF 30 F8 08 F9 F8
.:5E48 78 79 79 79 79 89 87 87
.:5E50 F7 07 78 08 F8 08 18 39
.:5E58 FF 08 F9 F9 F9 09 F9 39
.:5E60 F9 09 FF 18 F8 FF FF 30
.:5E68 FF F8 F8 87 78 09 F9 78
.:5E70 89 78 19 09 F9 09 C9 29
.:5E78 F8 87 89 89 F9 78 58 08
.:5E80 50 78 59 FD 19 F9 09 E9
.:5E88 19 F9 08 F9 38 FF 02 DF
.:5E90 18 F8 38 79 89 78 08 F8
.:5E98 29 F9 08 F8 09 E9 79 F7
.:5EA0 37 F7 09 89 89 F8 09 FF
.:5EA8 00 F8 09 F9 F9 F8 09 F9
.:5EB0 29 F9 29 F9 19 F8 00 F8
.:5EB8 F8 F9 79 89 79 79 03 E9
.:5EC0 48 E8 30 E8 09 F9 18 87
.:5EC8 89 89 87 F7 78 F8 28 C8
.:5ED0 20 F8 F9 F9 09 F9 F9 F9
.:5ED8 30 F9 F9 53 39 F8 D8 F8
.:5EE0 08 F9 09 F9 79 89 29 F9
.:5EE8 28 00 00 F7 87 F7 07 07
.:5EF0 37 87 89 89 08 F8 F8 F8
.:5EF8 0C FC C8 C9 C9 F9 F8 F9
.:5F00 FD F9 F9 39 39 53 FD FF
.:5F08 F9 79 F9 00 F9 79 78 28
.:5F10 F9 30 FF 78 89 87 87 87
.:5F18 78 78 78 79 F9 79 78 FF
.:5F20 CF 9C F9 9C F9 39 93 09
.:5F28 F9 09 FA FA FA 39 F9 F9
.:5F30 F9 F8 F9 39 F8 08 F9 39
.:5F38 7B 20 83 29 79 39 79 79
.:5F40 78 78 F7 39 89 09 F9 F9
.:5F48 FC 1C F9 DC FA 0F FA FF
.:5F50 FF 0F CF C9 FC 00 58 39
.:5F58 F9 00 F8 F8 F8 09 F8 38
.:5F60 FF 00 35 37 F7 F7 F7 39
.:5F68 FF FF FF FC CF FC FC FC
.:5F70 FC 0C CF 0F FF 0C CF 9F
.:5F78 F9 0C 4C 5F F5 CF FC CF
.:5F80 FC CF C9 FC 80 FF 07 C7
.:5F88 87 FC 7F F9 9F FC CF FC
.:5F90 CF FC 5F C5 9C F9 9F FF
.:5F98 FC FF CF FB 9F CF DB F7
.:5FA0 B7 7C 7C F5 9F F9 0C FF
.:5FA8 0C FF 0C FC F7 F7 09 F9
.:5FB0 7F FC C9 FC C9 FC C9 FC
.:5FB8 F9 FC FC DF F9 DF CF E8
.:5FC0 FC EC 3C EF 9F F7 17 E7
.:5FC8 A7 F7 FC FF F9 FF 2F CF
.:5FD0 2F FF FB FF 0F FF FF FF
.:5FD8 3C FF BC 3C 0F FF DF FF
.:5FE0 0C FF 9F F9 FC 4C 0F F8
.:5FE8 9C 9C 9C 9C 9C 9C 9C 9C
.:5FF0 9C 9C 9C 9C 9C 9C 9C 9C
.:5FF8 B1 F3 FF FD F2 23 A6 D0
.:6000 00 00 00 00 00 00 00 00
.:6008 00 00 00 00 00 00 00 00
.:6010 00 00 00 00 00 00 00 00
.:6018 00 00 00 00 00 00 00 00
.:6020 00 00 00 00 00 00 00 00
.:6028 00 00 00 00 00 00 00 00
.:6030 00 00 00 00 00 00 00 00
.:6038 00 00 00 00 00 00 00 00
.:6040 00 00 00 00 00 00 00 00
.:6048 00 00 00 00 00 00 00 00
.:6050 00 00 00 00 00 00 00 00
.:6058 00 00 00 00 00 00 00 00
.:6060 00 00 00 00 00 00 00 00
.:6068 00 00 00 00 00 00 00 00
.:6070 00 00 00 00 00 00 00 00
.:6078 00 00 00 00 00 00 00 00
.:6080 00 00 00 00 00 00 00 00
.:6088 00 00 00 00 00 00 00 00
.:6090 00 00 00 00 00 00 00 00
.:6098 00 00 00 00 00 00 00 00
.:60A0 00 00 00 00 00 00 00 00
.:60A8 00 00 00 00 00 00 00 00
.:60B0 00 00 00 00 00 00 00 00
.:60B8 00 00 00 00 00 00 00 00
.:61C0 00 00 00 00 00 00 00 00
.:61C8 00 00 00 00 00 00 00 00
.:61D0 00 00 00 00 00 00 00 00
.:61D8 00 00 00 00 00 00 00 00
.:61E0 00 00 00 00 00 00 00 00
.:61E8 00 00 00 00 00 00 00 00
.:61F0 00 00 00 00 00 00 00 00
.:61F8 00 00 00 00 00 00 00 00
.:6200 00 00 00 00 00 00 00 00
.:6208 00 00 00 00 00 00 00 00
.:6210 00 00 00 00 00 00 00 00
.:6218 00 00 00 00 00 00 00 00
.:6220 00 00 00 00 00 00 00 00
.:6228 00 00 00 00 00 00 00 00
.:6230 00 00 00 00 00 00 00 00
.:6238 00 00 00 00 00 00 00 00
.:6240 00 00 00 00 00 00 00 00
```

```
.:6248 00 00 00 00 00 00 00 00
.:6250 00 00 00 00 00 00 00 00
.:6258 00 00 00 00 00 00 00 00
.:6260 00 00 00 00 00 00 00 00
.:6268 00 00 00 00 00 00 00 00
.:6270 00 00 00 00 00 00 00 00
.:6278 00 00 00 00 00 00 00 00
.:6280 00 00 00 00 00 00 00 00
.:6288 00 00 00 00 00 00 00 00
.:6290 00 00 00 00 00 00 00 00
.:6298 00 00 00 00 00 00 00 00
.:62A0 00 00 00 00 00 00 00 00
.:62A8 00 00 00 00 00 00 00 00
.:62B0 00 00 00 00 00 00 00 00
.:62B8 00 00 00 00 00 00 00 00
.:62C0 00 00 00 00 00 00 00 00
.:62C8 00 00 00 00 00 00 00 00
.:62D0 00 00 00 00 00 00 00 0F
.:62D8 00 00 00 00 00 03 3F FF
.:62E0 00 00 00 00 0F FF FF FE
.:62E8 00 00 00 3F FF FF FA AA
.:62F0 00 00 3F FF FF FA AA AA
.:62F8 00 FF FF FF FA AA AA AA
.:6300 03 FF FF FF AA AA AA AA
.:6308 FF FF FF AA AA AA AA AA
.:6310 FF FF FF AA AA AA AA AA
.:6318 55 55 55 FF FF FF FF FF
.:6320 FF FF FF AA AA AA AA AA
.:6328 C0 FF FF FF AA AA AA AA
.:6330 00 FF FF FF AF AA AA AA
.:6338 00 00 FC FF FF AF AA AA
.:6340 00 00 00 FC FF FF AF AA
.:6348 00 00 00 00 F0 FF FF BF
.:6350 00 00 00 00 00 C0 FC FF
.:6358 00 00 00 00 00 00 00 F0
.:6360 00 00 00 00 00 00 00 00
.:6368 00 00 00 00 00 00 00 00
.:6370 00 00 00 00 00 00 00 00
.:6378 00 00 00 00 00 00 00 00
.:6380 00 00 00 00 00 00 00 00
.:6388 00 00 00 00 00 00 00 00
.:6390 00 00 00 00 00 00 00 00
.:6398 00 00 00 00 00 00 00 00
.:63A0 00 00 00 00 00 00 00 00
.:63A8 00 00 00 00 00 00 00 00
.:63B0 00 00 00 00 00 00 00 00
.:63B8 00 00 00 00 00 00 00 00
.:63C0 00 00 00 00 00 00 00 00
.:63C8 00 00 00 00 00 00 00 00
.:63D0 00 00 00 00 00 00 00 00
.:63D8 00 00 00 00 00 00 00 00
.:63E0 00 00 00 00 00 00 00 00
.:63E8 00 00 00 00 00 00 00 00
.:63F0 00 00 00 00 00 00 00 00
.:63F8 00 00 00 00 00 03 0F FF
.:6400 00 00 00 03 3F FF FF FE
.:6408 00 03 FF FF FF FA AA AA
.:6410 FF FF FF D5 55 55 55 56
.:6418 FF F5 55 55 55 55 6A AA
.:6420 AA AA AA AA AB FF FF FF
.:6428 AA AA AA AF FF FF FC 00
.:6430 AA AA BF FF FF F0 00 00
.:6438 AA BF FF FF F0 00 00 00
.:6440 AB FF FF FF 00 00 00 00
.:6448 FF FF FF 00 00 00 00 00
.:6450 FF FF FF 00 00 00 00 00
.:6458 FF FF FF 00 00 00 00 00
.:6460 FF FF FF 00 00 00 00 00
.:6468 EA FF FF FF 00 00 00 00
.:6470 AA FE FF FF 0F 00 00 00
.:6478 55 55 FD FF FF 0F 00 00
.:6480 AA AA AA FA FF FF 3F 00
.:6488 AA AA AA AA EA FF FF FF
.:6490 FF 5F 55 55 55 55 A9 AA
.:6498 FF FF FF 57 55 55 55 95
.:64A0 00 F0 FF FF FF AF AA AA
.:64A8 00 00 00 C0 FC FF FF BF
.:64B0 00 00 00 00 00 C0 F0 FF
.:64B8 00 00 00 00 00 00 00 00
.:64C0 00 00 00 00 00 00 00 00
.:64C8 00 00 00 00 00 00 00 00
.:64D0 00 00 00 00 00 00 00 00
.:64D8 00 00 00 00 00 00 00 00
.:64E0 00 00 00 00 00 00 00 00
.:64E8 00 00 00 00 00 00 00 00
.:64F0 00 00 00 00 00 00 00 00
.:64F8 00 00 00 00 00 00 00 00
.:6500 00 00 00 00 00 00 00 00
.:6508 00 00 00 00 00 00 00 00
.:6510 00 00 00 00 00 00 00 00
.:6518 00 00 00 00 00 00 00 00
.:6520 00 00 00 00 00 00 00 0F
.:6528 00 00 00 03 0F 3F FF FE
.:6530 03 0F FF FF FE FA AA AA
.:6538 FF FD F5 D5 55 55 56 5A
.:6540 AA AA AA AA AF BF FF FC
.:6548 AA AB AF FF FF FC C0 00
.:6550 AF FF FF FC C0 00 00 00
.:6558 FF FC 00 00 00 00 00 00
.:6560 C0 00 00 00 00 00 00 00
.:6568 00 00 00 00 00 00 00 00
.:6570 00 00 00 00 00 00 00 00
.:6578 00 00 00 00 00 00 00 00
.:6580 00 00 00 00 00 00 00 00
.:6588 00 00 00 00 00 00 00 00
.:6590 00 00 00 00 00 00 00 00
.:6598 00 00 00 00 00 00 00 00
.:65A0 00 00 00 00 00 00 00 00
.:65A8 00 00 00 00 00 00 00 00
.:65B0 00 00 00 00 00 00 00 00
.:65B8 00 00 00 00 00 00 00 00
.:65C0 00 00 00 00 00 00 00 00
.:65C8 03 00 00 00 00 00 00 00
.:65D0 FF 3F 00 00 00 00 00 00
.:65D8 F5 FF FF 3F 03 00 00 00
.:65E0 AA EA FA FF FF 3F 03 00
.:65E8 A9 AA AA AA FA FE FF 3F
.:65F0 FF 7F 5F 57 55 55 95 A5
```

```
.:65F8 C0 F0 FF FF BF AF AA AA
.:6600 00 00 00 C0 F0 FC FF BF
.:6608 00 00 00 00 00 00 00 F0
.:6610 00 00 00 00 00 00 00 00
.:6618 00 00 00 00 00 00 00 00
.:6620 00 00 00 00 00 00 00 00
.:6628 00 00 00 00 00 00 00 00
.:6630 00 00 00 00 00 00 00 00
.:6638 00 00 00 00 00 00 00 00
.:6640 00 00 00 00 00 00 00 00
.:6648 00 00 00 00 00 00 00 00
.:6650 00 00 00 00 00 00 03 0F
.:6658 00 00 03 0F 3F FF FF FE
.:6660 3F FF FF FE FA EA AA AA
.:6668 F5 D5 55 55 55 5A 6A AA
.:6670 AA AB AF BF FF FC F0 C0
.:6678 BF FF FC F0 C0 00 00 00
.:6680 F0 00 00 00 00 00 00 00
.:6688 00 00 00 00 00 00 00 00
.:6690 00 00 00 00 00 00 00 00
.:6698 00 00 00 00 00 00 00 00
.:66A0 00 00 00 00 00 00 00 00
.:66A8 00 00 00 00 00 00 00 00
.:66B0 00 00 00 00 00 00 00 00
.:66B8 00 00 00 00 00 00 00 00
.:66C0 00 00 00 00 00 00 00 00
.:66C8 00 00 00 00 00 00 00 00
.:66D0 00 00 00 00 00 00 00 00
.:66D8 00 00 00 00 00 00 00 00
.:66E0 00 00 00 00 00 00 00 00
.:66E8 00 00 00 00 00 00 00 00
.:66F0 00 00 00 00 00 00 00 00
.:66F8 00 00 00 00 00 00 00 00
.:6700 00 00 00 00 00 00 00 00
.:6708 00 00 00 00 00 00 00 00
.:6710 00 00 00 00 00 00 00 00
.:6718 00 00 00 00 00 00 00 00
.:6720 00 00 00 00 00 00 00 00
.:6728 0F 00 00 00 00 00 00 00
.:6730 FE FF 3F 0F 03 00 00 00
.:6738 AA EA FA FE FF 3F 0F 03
.:6740 5F 57 55 55 55 A5 A9 AA
.:6748 FC FF FF BF AF AB AA AA
.:6750 00 00 C0 F0 FC FF FF BF
.:6758 00 00 00 00 00 00 C0 F0
.:6760 00 00 00 00 00 00 00 00
.:6768 00 00 00 00 00 00 00 00
.:6770 00 00 00 00 00 00 00 00
.:6778 00 00 00 00 00 00 00 00
.:6780 00 00 00 00 F0 FF FE AA
.:6788 00 00 02 0A 2A AA FA 7F
.:6790 3F FF FF FE FA EA AA AA
.:6798 F5 D5 55 55 56 5A 6A AA
.:67A0 AB AF BF FF FF FC F0 C0
.:67A8 FF FC F0 C0 00 00 00 00
.:67B0 00 00 00 00 00 00 00 00
.:67B8 00 00 00 00 00 00 00 00
.:67C0 00 00 00 00 00 00 00 00
.:67C8 00 00 00 00 00 00 00 00
.:67D0 00 00 00 00 00 00 00 00
.:67D8 00 00 00 00 00 00 00 00
.:67E0 00 00 00 00 00 00 00 00
.:67E8 00 00 00 00 00 00 00 00
.:67F0 00 00 00 00 00 00 00 00
.:67F8 00 00 00 00 00 00 00 00
.:6800 00 00 00 00 00 00 00 00
.:6808 00 00 00 00 00 00 00 00
.:6810 00 00 00 00 00 00 00 00
.:6818 00 00 00 00 00 00 00 00
.:6820 00 00 00 00 00 00 00 00
.:6828 00 00 00 00 00 00 00 00
.:6830 00 00 00 00 00 00 00 00
.:6838 00 00 00 00 00 00 00 00
.:6840 00 00 00 00 00 00 00 00
.:6848 00 00 00 00 00 00 00 00
.:6850 00 00 00 00 00 00 00 00
.:6858 00 00 00 00 00 00 00 00
.:6860 00 00 00 00 00 00 00 00
.:6868 00 00 00 00 00 00 00 00
.:6870 00 00 00 00 00 00 00 00
.:6878 00 00 00 00 00 00 00 00
.:6880 FF 3F 0F 03 00 00 00 00
.:6888 7F 5F 57 55 55 15 05 01
.:6890 5F 57 55 55 95 95 A5 AA
.:6898 FC FF FF BF AF AB AA AA
.:68A0 00 00 C0 F0 FC FF FF FF
.:68A8 00 00 00 00 00 00 C0 F0
.:68B0 00 00 00 00 00 00 00 00
.:68B8 00 00 00 00 00 00 00 00
.:68C0 FF FF FF FF FF FF FF FF
.:68C8 AB AF AF AF AF AB BF AB
.:68D0 A9 A9 A5 95 D5 D5 F4 FC
.:68D8 FF FC F0 C0 C0 00 00 00
.:68E0 00 00 00 00 00 00 00 00
.:68E8 00 00 00 00 00 00 00 00
.:68F0 00 00 00 00 00 00 00 00
.:68F8 00 00 00 00 00 00 00 00
.:6900 00 00 00 00 00 00 00 00
.:6908 00 00 00 00 00 00 00 00
.:6910 00 00 00 00 00 00 00 00
.:6918 00 00 00 00 00 00 00 00
.:6920 00 00 00 00 00 00 00 00
.:6928 00 00 00 00 00 00 00 00
.:6930 00 00 00 00 00 00 00 00
.:6938 00 00 00 00 00 00 00 00
.:6940 00 00 00 00 00 00 00 00
.:6948 00 00 00 00 00 00 00 00
.:6950 00 00 00 00 00 00 00 00
.:6958 00 00 00 00 00 00 00 00
.:6960 00 00 00 00 00 00 00 00
.:6968 00 00 00 00 00 00 00 00
.:6970 00 00 00 00 00 00 00 00
.:6978 00 00 00 00 00 00 00 00
.:6980 00 00 00 00 00 00 00 00
.:6988 00 00 00 00 00 00 00 00
.:6990 00 00 00 00 00 00 00 00
.:6998 00 00 00 00 00 00 00 00
.:69A0 00 00 00 00 00 00 00 00
```

```
.:69A8 00 00 00 00 00 00 00 00
.:69B0 00 00 00 00 00 00 00 00
.:69B8 00 00 00 00 00 00 00 00
.:69C0 00 00 00 00 00 00 00 00
.:69C8 00 00 00 00 00 00 00 00
.:69D0 FF 3F 0F 03 00 00 00 00
.:69D8 FF 7F 5F 57 57 15 05 0A
.:69E0 BF AF AB AB AA AA A9 55
.:69E8 54 55 5F 5F 7E FE FE FA
.:69F0 00 00 00 FC AB BE EA EB
.:69F8 00 00 00 00 C0 BF AA EB
.:6A00 EA FF FF FF EB FF FF FF
.:6A08 AF AA AA AB AA AA BA AA
.:6A10 BA FF FF FF FF FF FF AF
.:6A18 FC BC BF AA AA AA AA AA
.:6A20 00 00 FF BF AA AA BA AA
.:6A28 00 00 FF AB BE AA AB BF
.:6A30 00 00 C0 F0 F0 B0 F0 B0
.:6A38 00 00 00 00 00 00 00 00
.:6A40 00 00 00 00 00 00 00 00
.:6A48 00 00 00 00 00 00 00 00
.:6A50 00 00 00 00 00 00 00 00
.:6A58 00 00 00 00 00 00 00 00
.:6A60 00 00 00 00 00 00 00 00
.:6A68 00 00 00 00 00 00 00 00
.:6A70 00 00 00 00 00 00 00 00
.:6A78 00 00 00 00 00 00 00 00
.:6A80 00 00 00 00 00 00 00 00
.:6A88 00 00 00 00 00 00 00 00
.:6A90 00 00 00 00 00 00 00 00
.:6A98 00 00 00 00 00 00 00 00
.:6AA0 00 00 00 00 00 00 00 00
.:6AA8 00 00 00 00 00 00 00 00
.:6AB0 00 00 00 00 00 00 00 00
.:6AB8 00 00 00 00 00 00 00 00
.:6AC0 00 00 00 00 00 00 00 00
.:6AC8 00 00 00 00 00 00 00 00
.:6AD0 00 00 00 00 00 00 00 00
.:6AD8 00 00 00 00 00 00 00 00
.:6AE0 00 00 00 00 00 00 00 00
.:6AE8 00 00 00 00 00 00 00 00
.:6AF0 00 00 00 00 00 00 00 00
.:6AF8 00 00 00 00 00 00 00 03
.:6B00 00 00 00 00 00 00 0F FA
.:6B08 00 00 00 03 0F 0F FF AA
.:6B10 00 00 FF FF FA FF F5 55
.:6B18 0F 3F FF AA AF FD 55 55
.:6B20 FF FE AA EA FF 55 FF FE
.:6B28 FF F5 FF 55 6A A5 57 FF
.:6B30 AA 6A A9 55 FF 55 A6 A9
.:6B38 FF FF 55 AA 95 5F FF FF
.:6B40 FF FF FF FF FF FF FF FF
.:6B48 FF EB FF FF FF FF FF FF
.:6B50 FF FF FF FF FF FF AF FF
.:6B58 FF FF FF FF FF FF FF AB
.:6B60 FF FF FF FF FF AA FF FF
.:6B68 FF FF FF FE AA BE FF FF
.:6B70 E8 E8 F8 A8 F8 E8 E8 FA
.:6B78 00 00 00 00 00 00 00 00
.:6B80 00 00 00 00 00 00 00 00
.:6B88 00 00 00 00 00 00 00 00
.:6B90 00 00 00 00 00 00 00 00
.:6B98 00 00 00 00 00 00 00 00
.:6BA0 00 00 00 00 00 00 00 00
.:6BA8 00 00 00 00 00 00 00 00
.:6BB0 00 00 00 00 00 00 00 00
.:6BB8 00 00 00 00 00 00 00 00
.:6BC0 00 00 00 00 00 00 00 00
.:6BC8 00 00 00 00 00 00 00 00
.:6BD0 00 00 00 00 00 00 00 00
.:6BD8 00 00 00 00 00 00 00 00
.:6BE0 00 00 00 00 00 00 00 00
.:6BE8 00 00 00 00 00 00 00 00
.:6BF0 00 00 00 00 00 00 00 00
.:6BF8 00 00 00 00 00 00 00 00
.:6C00 00 00 00 00 00 00 00 00
.:6C08 00 00 00 00 00 00 00 00
.:6C10 00 00 00 00 00 00 00 00
.:6C18 00 00 00 00 00 00 00 00
.:6C20 00 00 00 00 00 00 00 00
.:6C28 00 00 00 00 00 00 00 00
.:6C30 00 00 00 00 00 03 0F 0F
.:6C38 0E 3A EA EA FA FA FE FF
.:6C40 FF FF AA AA EA FA FF FF
.:6C48 FF AA A9 AA A5 95 AA FE
.:6C50 FE AA 56 95 55 56 A5 AA
.:6C58 FF FF FF AA AA EB AA AA
.:6C60 FA FA EA AF FF FF FF AA
.:6C68 FF FF AF BF AF FF BF FE
.:6C70 FF AF BF FA FF FE FF BF
.:6C78 EB FF FF FF FF AF FF FF
.:6C80 FF FF FF FF FF FF EF FF
.:6C88 FF FF FF FF FF FF FF FF
.:6C90 FF FF FF FF FE FF FF FF
.:6C98 FF FF FF FF AF FF FF FF
.:6CA0 FF FF AB FF FA FF FF FF
.:6CA8 FF FF FF FF FF FF FF FF
.:6CB0 FC C0 C0 F0 F0 AF FA EB
.:6CB8 00 00 00 00 00 00 F0 FF
.:6CC0 00 00 00 00 00 00 00 00
.:6CC8 00 00 00 00 00 00 00 00
.:6CD0 00 00 00 00 00 00 00 00
.:6CD8 00 00 00 00 00 00 00 00
.:6CE0 00 00 00 00 00 00 00 00
.:6CE8 00 00 00 00 00 00 00 00
.:6CF0 00 00 00 00 00 00 00 00
.:6CF8 00 00 00 00 00 00 00 00
.:6D00 00 00 00 00 00 00 00 00
.:6D08 00 00 00 00 00 00 00 00
.:6D10 00 00 00 00 00 00 00 00
.:6D18 00 00 00 00 00 00 00 00
.:6D20 00 00 00 00 00 00 00 00
.:6D28 00 00 00 00 00 00 00 00
.:6D30 00 00 00 00 00 00 00 00
.:6D38 00 00 00 00 00 00 00 00
.:6D40 00 00 00 00 00 00 00 00
.:6D48 00 00 00 00 00 00 00 00
```

```
.:6D50 00 00 00 00 00 00 00 00
.:6D58 00 00 00 00 00 00 00 00
.:6D60 00 00 00 00 00 00 00 00
.:6D68 00 00 00 00 00 03 03 0F
.:6D70 3E 3E 3E 3F 3E FA AA EA
.:6D78 FE EA FE BE FE FF AB AA
.:6D80 FA FF FF FF FF 5F 55 55
.:6D88 FF AF AA AA AA 55 55 55
.:6D90 FE FF BF AA AA AA 55 55
.:6D98 FF AB AA EA FF FF 7F 55
.:6DA0 FA FF 57 55 D5 FF FF AA
.:6DA8 FF AF AA 56 55 95 AA EA
.:6DB0 FF FF FF AA 5B 56 55 95
.:6DB8 FF DF FF FF FF 7F 9D A5
.:6DC0 FF FF FF FF FF FF FF FA
.:6DC8 FF FF FF FF FE EA AA BF
.:6DD0 FF FF FA AA BF BF FF FF
.:6DD8 FA EA AA FF FF AF FF FF
.:6DE0 FF AF AA FE FE FE AB FF
.:6DE8 FA AA AA AA AA FB FA FA
.:6DF0 FF FF AB FF FF FF FF FF
.:6DF8 EA FA FF FF FF FF FA FF
.:6E00 FF EA AB AA AA AF EA AA
.:6E08 F0 B0 FC BC BC AC AC AC
.:6E10 00 00 00 00 00 00 00 00
.:6E18 00 00 00 00 00 00 00 00
.:6E20 00 00 00 00 00 00 00 00
.:6E28 00 00 00 00 00 00 00 00
.:6E30 00 00 00 00 00 00 00 00
.:6E38 00 00 00 00 00 00 00 00
.:6E40 00 00 00 00 00 00 00 00
.:6E48 00 00 00 00 00 00 00 00
.:6E50 00 00 00 00 00 00 00 00
.:6E58 00 00 00 00 00 00 00 00
.:6E60 00 00 00 00 00 00 00 00
.:6E68 00 00 00 00 00 00 00 00
.:6E70 00 00 00 00 00 00 00 03
.:6E78 00 00 00 00 3F FA FB FD
.:6E80 00 00 0F FF FE FA AB AD
.:6E88 C0 F0 F0 FC BF F5 75 7F
.:6E90 00 00 00 00 C0 FF AA AF
.:6E98 00 00 00 00 00 00 C0 FF
.:6EA0 00 00 00 00 00 0F 0E 0E
.:6EA8 0E 3F 3A 3A FA EA AB BF
.:6EB0 FF AB FF FA FF FF AF FF
.:6EB8 FF FF FF BF FF FF FF FA
.:6EC0 FF FF FF AF FF FF FF FF
.:6EC8 FE AA AA AA AA BE AA AA
.:6ED0 FF AF FF FF FF FF FF AB
.:6ED8 FF FF FA FF EB FF FF FF
.:6EE0 FA FA FF FF FF EA EA E9
.:6EE8 FF FF AA BF BF F5 D5 57
.:6EF0 FE FE FA EA AB AF BF F5
.:6EF8 F9 F9 E5 A5 A9 95 5A AA
.:6F00 FE FE AA FF FA FA FF FF
.:6F08 FF FF FA FF FF FF BF FF
.:6F10 FA FF BF FF FF FF FF FF
.:6F18 EA AA AA AA AA AA AA AA
.:6F20 FA AA AA AA AA AA BA AA
.:6F28 FF FF FE FE FA FA EA AB
.:6F30 EA EB EA EA FA FA BB BE
.:6F38 FF BF FF FF AF FF BF FF
.:6F40 FA AA AA FE AA AA AA AA
.:6F48 FE AA AA FE FF FF FF FE
.:6F50 00 00 C0 C0 C0 C0 C0 C0
.:6F58 00 00 00 00 00 00 00 00
.:6F60 00 00 00 00 00 00 00 00
.:6F68 00 00 00 00 00 00 00 00
.:6F70 00 00 00 00 00 00 00 00
.:6F78 00 00 00 00 00 00 00 00
.:6F80 00 00 00 00 00 00 00 00
.:6F88 00 00 00 00 00 00 00 00
.:6F90 00 00 00 00 00 00 00 00
.:6F98 00 00 00 00 00 00 00 00
.:6FA0 00 00 00 00 00 00 00 00
.:6FA8 00 00 00 00 00 00 00 03
.:6FB0 03 0F 0F 3E 3E FF FF FA
.:6FB8 FB AA AA BF AA EA EA AA
.:6FC0 FE BF 6B 56 55 55 AA 55
.:6FC8 FE AB 56 95 E9 FE FF FF
.:6FD0 EA AA EA 7E 57 D5 BD AB
.:6FD8 FE AA AA AF FA 5F 55 F5
.:6FE0 FE AE BE FE BE BE EA EA
.:6FE8 FF FF FF FF FF FF FA FA
.:6FF0 FA FA FF FF AF FF AA AA
.:6FF8 FF FF AA AA AA AB FF D5
.:7000 FF FF FF FA AA A9 55 55
.:7008 FF FE FE AA A5 55 55 9A
.:7010 FF AA AA A5 55 56 AA AA
.:7018 EA AA A5 55 56 AA AA AA
.:7020 FA AA AB BF FF FF FF F5
.:7028 FA AA A9 A9 AA A9 AA 6A
.:7030 FE FF FE BF FB BB FF FF
.:7038 FF AB FA AB BF BA AA AA
.:7040 FA AA FF FA FB FE FE FE
.:7048 FF FF FF FF AF FF EA FF
.:7050 FF FE FF EB BF EF FB AF
.:7058 FF BA FF EB EE EB EE EB
.:7060 FF FB FF FF FF EA FF FF
.:7068 FF EF FF FF FF FF FF EA
.:7070 FB FB EB EA AA AF EF FF
.:7078 FF FF FF FF AF EA FE FE
.:7080 FF FF FF FF FF FF FF BF
.:7088 FE FF FF FF FA FF FF FF
.:7090 F0 B0 BC AC FF AB AB AB
.:7098 00 00 00 00 00 00 00 00
.:70A0 00 00 00 00 00 00 00 00
.:70A8 00 00 00 00 00 00 00 00
.:70B0 00 00 00 00 00 00 00 00
.:70B8 00 00 00 00 00 00 00 00
.:70C0 00 00 00 00 00 00 00 00
.:70C8 00 00 00 00 00 00 00 00
.:70D0 00 00 00 00 00 00 00 00
.:70D8 00 00 00 00 00 00 00 00
.:70E0 00 00 00 00 00 00 3F FF
.:70E8 03 0F 0F 0F 3E 3E FE FE
.:70F0 FA FA FA AA AF AA AA FA
.:70F8 EB FF FF FF BF FF FF AF
```

```
.:7100 FF FF FF EA AF FF FF FF
.:7108 FF FF FF BF FF FF FF FF
.:7110 FF FF FF FF FF FF FF FF
.:7118 FA FE FA FB FB FA EA EB
.:7120 EA EB AB AF FD D5 D5 F5
.:7128 EA AA A9 95 55 56 5A 5B
.:7130 FE EA AA AB BF FF FF 7F
.:7138 FF FE EA 96 AA AA AA AA
.:7140 FA AA AA AA AA AA A5 AA
.:7148 FF FF FF FF FF FF BF FE
.:7150 FF FF FF FF FF FF FF BF
.:7158 FF FF FF FF AB FF FF FF
.:7160 FA FF FF FF FF FF FF AA
.:7168 FE FF FF FF AF EA FF FF
.:7170 EA AA AA AA AA EA AA AA
.:7178 FF AB FF FF EF EA FF FF
.:7180 FE FE FE FA FA FE FF FF
.:7188 FF FF AB FF FF FF AF BF
.:7190 FF BE FF FF FF FF AB FF
.:7198 FF BF FF FF FF FF FF FF
.:71A0 EB FF FF FF FF FF FF FF
.:71A8 FF FF FF FA EF FF FF FF
.:71B0 FF FE FF FF EA FF FF FF
.:71B8 FE AF FF FF FE FF FF FF
.:71C0 FA BE BF AF FA AA AA AA
.:71C8 FF FF AA FE FF FF FF FF
.:71D0 FE FE FF BF AF EA EA FF
.:71D8 00 C0 F0 B0 BF EB EB EB
.:71E0 00 00 00 00 00 00 00 00
.:71E8 00 00 00 00 00 00 00 00
.:71F0 00 00 00 00 00 00 00 00
.:71F8 00 00 00 00 00 00 00 00
.:7200 00 00 00 00 00 00 00 00
.:7208 00 00 00 00 00 00 00 00
.:7210 00 00 00 00 00 00 00 03
.:7218 0F 0F 0F 0F 3F 3F FF FF
.:7220 EA FE EA AF AA FA AA EA
.:7228 FE BE BA BA AA AA FA AA
.:7230 FF FF EB FF FF FF EB FF
.:7238 FA AA AA AB BF AA AA AA
.:7240 FA AA AA EA FA AA AF F5
.:7248 FF FF FF FF FA A5 56 6B
.:7250 FF FF FA A5 56 6A BE FA
.:7258 EB EF AF 6F AB AB AF AE
.:7260 E9 AA AA AF EF EE AE AF
.:7268 FE FF BF AF 6B AA AA 55
.:7270 FF FF BF AF AA EA FF 55
.:7278 FF FF FF FF AA AA FF 7F
.:7280 FF FF FF FF AF AA AA FF
.:7288 FF FF FF FF FF AF AA AA
.:7290 FF FE AA FF FF FF FF 55
.:7298 FF BF AB FF FF FF FF FF
.:72A0 FF FF FA FF FF FF FA FF
.:72A8 EB FF AF AB FF FF FF FF
.:72B0 FF FF EB FF FF FF AA FF
.:72B8 EF FA BF AA AA AA AA AA
.:72C0 FF FF FF FF FF FF FF FF
.:72C8 FA EA EA FA EA AA AA AA
.:72D0 FF FF FF FF FF FF FA FF
.:72D8 FF FF FF FF AF FF BF FF
.:72E0 FF FF EA FF FF FF FF AB
.:72E8 FF FF FF FF AB FF FF FF
.:72F0 FF FF FF FF FF FF FF AB
.:72F8 FF FF FF EF FB FB FE FE
.:7300 FF FF FF FF FF FF AF AA
.:7308 FF FF FF FF FF FF AB AA
.:7310 FF FF FF FF FF FF FF FF
.:7318 FB FA FA FE FE BE AE AF
.:7320 00 C0 C0 F0 B0 B0 AC AC
.:7328 00 00 00 00 00 00 00 00
.:7330 00 00 00 00 00 00 00 00
.:7338 00 00 00 00 00 00 00 00
.:7340 00 00 00 00 00 00 00 00
.:7348 00 00 00 00 0F 0F 3F FB
.:7350 0F 0F FF FF FE FA AA AA
.:7358 FF FE FA EB ED EB 7A 5E
.:7360 7F 7A EA AA AA AA EA 7E
.:7368 FF BF BF AB AB AA FA FA
.:7370 FE AF FF FF FF FA FA AA
.:7378 EB AD AB AA AA AA AA AA
.:7380 FE F9 FE AF 5A 55 55 55
.:7388 EA AB EB 7F 57 FF AB AF
.:7390 EB AB AF BE FF AB AA FF
.:7398 FF FF FA BF FE FE FA FE
.:73A0 EF AF AF AF AF FF FF FF
.:73A8 FF FF FA FF FF FF AF FF
.:73B0 FF FF AF FF FF FF FF FF
.:73B8 FE FF EB FF FF FF FF FA
.:73C0 FF AF AF AA AA AA AA AF
.:73C8 FA FF 57 55 55 7D 55 5F
.:73D0 FF AB AA 69 55 A5 5A AA
.:73D8 FF FF AA AA 5A AA AA FF
.:73E0 EA FA FE BF AF BF FE FA
.:73E8 FF FE FF FF 7F FF FF FF
.:73F0 FF AB FF FF FF FF AB FF
.:73F8 FA FF FF FF FF FF FF FF
.:7400 FF FF FF FF FF FF FF FF
.:7408 FA AA FA FA FE FF FF FF
.:7410 FF FF EA FF FF FF AF AF
.:7418 FF FF FF FF FF FF FF FF
.:7420 FF FF FE FF FF FF FF FF
.:7428 FF FF BF FF FF FF FF FF
.:7430 FF FF FF FF AF FF FF FF
.:7438 FF FF FF FF FF FF AB FF
.:7440 EE FF FF FF FF FF FF FF
.:7448 FA FE FF FF FF FF FF EB
.:7450 EA EA EA EA FA FA AA EB
.:7458 FA FA FA FF FF FF BF FF
.:7460 EC FC FC FC BC BF AF AF
.:7468 00 00 00 00 00 00 00 00
.:7470 00 00 00 00 00 00 00 00
.:7478 00 00 00 00 00 03 03 03
.:7480 03 0E 3E FE FE EE EA EA
.:7488 FB BB AB AA AA AA AA FA
.:7490 FE EA AA AA AA FE AA AB
.:7498 FE FF AB FF FF FE E9 95
.:74A0 FE BF 6B 56 6B BF FA A9
.:74A8 EF 7F 5F 5E 7A EA AA AA
```

```
.:74B0 FF FF FF AE AE AA AA AA
.:74B8 FA FA FA FE FE FF BF BF
.:74C0 FF FF FF FF FF BF BF AF
.:74C8 FA FA EA EA EA AA AB BF
.:74D0 FE EA EB EB EB EF AF BF
.:74D8 FE FE FE FE FE FE FE FE
.:74E0 FF FF FF FF FF EB FF FF
.:74E8 FF FF FF FE FA FF AA FF
.:74F0 EA EA EA AA A9 A5 A9 EA
.:74F8 FF FF FE AA AA BF AA AA
.:7500 FF FA AA AA BF FF FF AB
.:7508 FE AA AA FF FF FF FF FF
.:7510 FF FF EA AA AA AA AA AA
.:7518 FF FA AA AA AA AA AA A9
.:7520 FA AA AA A9 AA 96 6A AA
.:7528 FA FF FF AF FF FA FF FF
.:7530 EA AA AA AA AA AA AA AA
.:7538 FF FF FF FF FF FF FF FF
.:7540 FF FF FF FF FF FF FF FF
.:7548 FF FF FF FE FF FF FF FF
.:7550 FF FF FF FF AF AF BF BE
.:7558 FF FF FF AB FF FF FE BF
.:7560 FF FF FF FF FF FF FB AF
.:7568 FA AA AA AA AA AA AB BE
.:7570 FF FF AB FF FF FF BF FB
.:7578 FF FF FF FF FF FF FF BB
.:7580 FF FF FF FE FF FF FF EB
.:7588 FF FF FF AF FF FF FF EB
.:7590 FE FE FF FF FF FF FF EB
.:7598 FB AA FE FF FF FF FF EE
.:75A0 FE FF AF AF EB FA FB EB
.:75A8 C0 C0 F0 F0 B0 B0 B0 F0
.:75B0 00 00 00 00 00 00 00 00
.:75B8 00 00 03 03 00 0F 0E 0F
.:75C0 FA 3A FF EA FF FA AA AA
.:75C8 FF FF AE FF FF FF FF AF
.:75D0 E9 95 5A 95 E9 FE FF FF
.:75D8 FA A5 55 A5 FA BF 6B 56
.:75E0 FF FF FF FF FF AB 56 A5
.:75E8 FF FF FF FF FF FF BF 6F
.:75F0 FF FF FF FF FF FF FF FF
.:75F8 FA FA FF FF FF FF FF FA
.:7600 FF FF FF AF AF AF BF FF
.:7608 FB FA FA FF EB AF FF AF
.:7610 FF FF FF FF FF FF FF FF
.:7618 FE FA FA FA FA EA EA FA
.:7620 FA FF FF AB FF FF FF FF
.:7628 FF AF FF FF FF FE FA EA
.:7630 FF BF FF FF FF FF FF FF
.:7638 FE FF FF FF FD D5 FF FD
.:7640 FF AF AA AA AA AA A5 AA
.:7648 FF FF AB AA AA AA A5 AA
.:7650 FF AA AA FF 5F FF FF FF
.:7658 FF BF AF AB EA FA EA AA
.:7660 FF FA FF FF FF 7F FF FF
.:7668 FF BF FF FF FF FF FF FF
.:7670 FF FF FF FF EA FF FF FF
.:7678 FF FF FF FF FF FF FF AB
.:7680 FF FF FF FF FF FF FF FF
.:7688 EB EA FA FA FF FF FF FF
.:7690 FF FF FF FF AF AB AA AB
.:7698 FF FF FF FF FF FF BF FF
.:76A0 FF FF FE BB FF FF FF FF
.:76A8 FF FF BF FF FF FF FF FF
.:76B0 FF FF FF FF FF FF FF FA
.:76B8 FF FF FF FF AB FF FF FF
.:76C0 FF FF FF FF FF FF FF FF
.:76C8 BF FF FF AF FF FF FF EB
.:76D0 FE FF FF FF FF FF FF FF
.:76D8 FF FF FF FF FB FA FA FA
.:76E0 FB FB FF FF FF FF 7F 5F
.:76E8 FC BB AA AA AA AA AA AA
.:76F0 00 C0 C0 C3 FF AB AB AA
.:76F8 0F 3A FF FA AA FE FF AF
.:7700 FE AA AA AE EF BF AB FA
.:7708 FF FF FF AF AA AF BF FF
.:7710 FF FA FF FF FF FE FF FF
.:7718 FF BF FF FF FF AA EB FF
.:7720 FA FE F9 E5 E5 F9 FE FF
.:7728 F9 F9 E5 95 95 E5 F9 BE
.:7730 FF FF FF FF FE FE FE FE
.:7738 FA FA EA AA AA AA AB BF
.:7740 FF FF FA FF FF AF AF AF
.:7748 FF FF FF FF FF FF FF FF
.:7750 FF FF FF FF FF FF FF FF
.:7758 FF FF FF FF FF FF FA EA
.:7760 FF FF FF FF FA AA AA AB
.:7768 FF FF FF FF AA AA AA FF
.:7770 FF FF FE AA AA AA AB FF
.:7778 FF FF AA AA AA AF FF FF
.:7780 FF FA AA AA AA FF FF FF
.:7788 FF AA AA AA BF FF F5 55
.:7790 FF FF FF FE AA AA 55 55
.:7798 FE FA AA AA AA A5 55 55
.:77A0 FF FF FF EA FF AA AA AA
.:77A8 FF FF FF AB EB AB AB AA
.:77B0 FF FF FF FF FF FF FE FF
.:77B8 FF FF FF FF AF FF AF FF
.:77C0 FF FF FF FF FF FF AF AF
.:77C8 FF FF FF FF FF FF AF AF
.:77D0 FA FF FF FF FF FF 5F 5F
.:77D8 FF FF AA AA AA AA 5A 5A
.:77E0 FF FF AF AA AA AA 5A 5A
.:77E8 FF FF FF FF AA AA FF AA
.:77F0 FA AA AA AA AA FA AA AA
.:77F8 FF FF FF EA FF FF FF FF
.:7800 FF FF FF FF FF FF FF FF
.:7808 FE FF FF FF FF FF FF FF
.:7810 AF FF FF FF FF FF AA FA
.:7818 F5 F7 F7 F7 F7 F6 A6 A6
.:7820 57 57 D5 FD FF FF EF AB
.:7828 FF FF FF BF FF FF FF FF
.:7830 FF FF FF FF FF FF FF FF
.:7838 FF FF FF FF FF FF FF FF
.:7840 FF AF AB AB EF EF EA FA
.:7848 FF FA FA FF FF FF BF FF
.:7850 FF FF AF FF FF FF AF FF
.:7858 FF FF FF FF FF FF FF FF
```

```
.:7860 FF FF FF FF FF FF FE FA
.:7868 E5 F9 FE FF EA EA AA AA
.:7870 FB 5F DF BE FE EA EB AA
.:7878 FF EA AA AF AA AA FB AF
.:7880 FA FA FF FA FA FE AA AA
.:7888 FF FF FF FF FF FF FF FF
.:7890 FF FF FF FF FF FF FF FF
.:7898 D5 F5 FF EF EA FA FE AB
.:78A0 FA FF FF AF AA 6A 55 95
.:78A8 FF FF AA AA EA FF 55 55
.:78B0 FF FF BF AA AA FF F5 55
.:78B8 FF FF FF BF AA FF 57 F5
.:78C0 FE FF FF FF 7F 55 FF AB
.:78C8 AA FE EA FF FF 7F 55 FF
.:78D0 FA AA AA EA FF FF 7F 55
.:78D8 FF FF BF FF FF AA AA 55
.:78E0 FF AF FF AF FF FF AA AA
.:78E8 FF FF FF FF FF FF FF AF
.:78F0 FF BF BF BF AA AA AA AA
.:78F8 FF FF FF FF FF FF FF FF
.:7900 5A AA A5 6B A5 AA F5 FA
.:7908 FA AF 55 55 FF FA FF AF
.:7910 55 F5 5A FA 5F 5F F5 55
.:7918 FF AF 5F 5A FF FB AF FA
.:7920 DA FA DA FA DA FA AA AA
.:7928 A6 A5 A5 A5 A6 A6 A2 80
.:7930 7F 5F 95 A5 69 56 55 55
.:7938 FF FF AF FA BF AB BF AF
.:7940 FF FF FF FF EA FF AA FA
.:7948 FF FF AA FE AA FA AA FE
.:7950 FF FF FE F2 E0 A2 F5 F5
.:7958 C0 2A 20 02 A0 80 55 7F
.:7960 FF 3F AF 83 00 28 57 57
.:7968 FF AB AF AF AA 2A AA AA
.:7970 FF FF FF FF FF BF EF EF
.:7978 FF FF FF FE FA FE FF EA
.:7980 FA EA AA AA AA AA AA AA
.:7988 FA EA AA FE AA EA FE AA
.:7990 FF FA EA EA EA EA AA AA
.:7998 FE AA AA AA AF AA AF BF
.:79A0 FF FF FE AA AA EB AA AA
.:79A8 EA FA AA AA AA FA AF FF
.:79B0 FF EA AA EA FA AA AA AA
.:79B8 FA AA AA AA AA AA AA AA
.:79C0 FF FF FF FF FF FF FF FF
.:79C8 FF FF FF FF FF FF FF FF
.:79D0 FD FD FF FF FF BF AF AB
.:79D8 FA FF AF BF FE AE BE AE
.:79E0 FA AA FF FF FE FF BE BA
.:79E8 FA FF FF BF BF BE BF AA
.:79F0 FF FB AB FB BA AA AA A9
.:79F8 FF EF EF AE AA A5 55 55
.:7A00 FA AA AF FD 55 57 5F 7F
.:7A08 AB FF F5 57 7F FF FF FF
.:7A10 FF FA AB FF FF FF FF FF
.:7A18 FF FE AA AA AA AA AA AA
.:7A20 EA A9 A9 AA AA AA AA AA
.:7A28 FF FF FF FF AA AA AA AA
.:7A30 FF FF FF FF FF FF AF AF
.:7A38 FF FE FE FA FA FA AA AA
.:7A40 FA F5 FA F5 FA F5 FA F5
.:7A48 FA AF FA AF FA AF FA AF
.:7A50 5F A5 AF A5 AF F5 55 DF
.:7A58 FF BA EF BA FF AA FF FA
.:7A60 FF FE F0 C0 02 FA F5 F5
.:7A68 0F C0 F0 03 C0 00 AA AA
.:7A70 3F 0F 2B 80 28 00 55 55
.:7A78 FF FF FF FF 3F FF F5 F5
.:7A80 FF FF FF FF FF FF F5 F5
.:7A88 FF FF FF FF FF FF FA FA
.:7A90 FA F5 FA F5 FA F5 FA F5
.:7A98 EA D5 5D EA EA F5 5F F5
.:7AA0 57 6B 97 6B 97 6B 97 6B
.:7AA8 FF FF FF FF FF FF FF FF
.:7AB0 FF FF AF FF AB FF FF FF
.:7AB8 EB AA AA AA AA AA AA AA
.:7AC0 FF BE AA AA FF AA AA AA
.:7AC8 FF FF FF FF FF FF FF FF
.:7AD0 FA FA FA FF FF FF FF FF
.:7AD8 FF FF FF FE FF FE EA AA
.:7AE0 FA EA AA AA FA AA AA AF
.:7AE8 FF FF FF EA FF AB FF FE
.:7AF0 FF FA AA AA AB AB AA EB
.:7AF8 FF FF BF BF FF FF FF FF
.:7B00 FF FF FF FF FF FF FF FF
.:7B08 FF FF FF FF FF FC F0 C0
.:7B10 55 7F 7F 7F 3F 3F 0F 03
.:7B18 FF 7F FF FF FF FF FE EA
.:7B20 FF FF FF FF FE AA AA AA
.:7B28 FF FF FE EA AA AB BF FF
.:7B30 FA AA AA AA AF FF FF FF
.:7B38 FF FE EA AA AA AA AA AA
.:7B40 FF FF FF FF FF FF FF FF
.:7B48 FF FF FF FF FF FF FF FF
.:7B50 FF FF FF FF FF FF FF FF
.:7B58 FF FF AF 5F AA 55 A5 55
.:7B60 FF FF AF 5F AA 65 AA 5A
.:7B68 FF FF AF 5F 5A 69 55 A5
.:7B70 FF FF 5F AF 5A 65 A6 59
.:7B78 FF FF AF 5F 5F 9F 5F AF
.:7B80 FA F5 FA F5 FA F5 FA F5
.:7B88 FA AF FA AF FA AF FA AF
.:7B90 FA F5 F6 FA 59 A6 5A F5
.:7B98 FA AF AA FF EA AF FA AF
.:7BA0 55 F5 5F F5 D5 7F D5 7F
.:7BA8 FA AF FA FF AF FA AE FA
.:7BB0 AA 5A A5 59 AA 9F 7F FF
.:7BB8 FA AB AF EA 55 55 55 55
.:7BC0 7D D5 5F F5 55 A5 AF AA
.:7BC8 EB AE FF AA FF AF BE AB
.:7BD0 FF BA EF FA AF FA EF FA
.:7BD8 FA AF FA AF FA AF FA A9
.:7BE0 D7 57 FD 55 59 68 A2 82
.:7BE8 EB D7 AA 5A A5 5A 25 49
.:7BF0 EB D7 55 A5 5A A5 55 9A
.:7BF8 EB D7 AA 59 AA 5A A5 AA
.:7C00 EB D7 55 95 A5 5A 55 A9
.:7C08 EB D7 A5 5A A5 96 AA 56
```

```
.:7C10 AA AA EA EA D6 DE D7 7D
.:7C18 FF FF FF FF FF FF 7F BF
.:7C20 57 55 FD FF FF FF FF FF
.:7C28 FF FF FF FF FF FF FF FF
.:7C30 FF FF FF FF FF FF FF FA
.:7C38 FF FF FF FF FF FA AA AA
.:7C40 FF FC FF FE EA AA AA AA
.:7C48 00 00 D6 DA D5 E9 E5 E9
.:7C50 00 00 E9 BD A9 BD A9 FD
.:7C58 F5 35 F5 F5 F5 E5 A5 A5
.:7C60 FA FA AA AA AA A9 A5 95
.:7C68 FA F5 56 A5 56 5A D5 F5
.:7C70 FA F5 6A A5 6A A5 5B AF
.:7C78 FA F5 A5 55 69 D5 FA FE
.:7C80 FA F5 AA 5A 96 5A AA 6B
.:7C88 FA F5 A5 5A A5 75 FE FF
.:7C90 FA FA AA AA A9 A5 A5 94
.:7C98 FF FB EA AA AA 88 22 28
.:7CA0 FF FA AF FA AB B9 A5 95
.:7CA8 FD 55 F7 5F A5 A9 AA AA
.:7CB0 FA AF EA AF AA EF BA 6F
.:7CB8 FF FF FF FF FF FF FF FF
.:7CC0 FA F5 FA F5 FA F5 FA F5
.:7CC8 5F F5 5F F5 5F F5 5F F5
.:7CD0 FA F5 F9 5A A5 5A A5 5A
.:7CD8 F5 5F D5 5F F5 5F 5F F5
.:7CE0 FA AF FA AF FA AF BA AD
.:7CE8 5B AF BF 7F FF FF FF FF
.:7CF0 FF FF FF FF FF FF FF FF
.:7CF8 FF FF FF FF FF FF FF FA
.:7D00 FF FF FF FF FF FA AA AA
.:7D08 AA FA FA FD 55 55 55 55
.:7D10 FF EA FF 7A 6F 7A 7E 7A
.:7D18 F6 5A EA 5A E9 5F F5 5F
.:7D20 F0 0C FC 03 F2 BA 9A 99
.:7D28 01 F1 0C FF FD 65 95 55
.:7D30 FF AF FA FF AF FA FE AF
.:7D38 FD 55 F5 5F 57 F5 55 5F
.:7D40 FF AF FA AF BA FA AF FA
.:7D48 57 F5 5F F5 5F 57 F5 5F
.:7D50 FB AA FF FA AF FE EB BE
.:7D58 55 6B A5 5A A5 55 5A A5
.:7D60 FF AA 6A 6A 5A 5F 55 55
.:7D68 FE AA AA AA AF FF FF FF
.:7D70 FF FF FF FA AA AA AA AA
.:7D78 FF FE AA AA AA A9 A9 95
.:7D80 5F FE FE EA EA AA AA AA
.:7D88 E5 56 A5 5A 65 55 A5 5A
.:7D90 FE 7F FF F5 5F F5 5F 55
.:7D98 F5 96 55 A5 6A 5A A5 5A
.:7DA0 EA 6A EA EA 6A EA FD D5
.:7DA8 FA F5 F5 FA F5 FA A5 5A
.:7DB0 FA 5A FA 7A 5A FA F5 FF
.:7DB8 FD FD FD FD FD FE 5A A5
.:7DC0 7E F6 7E 56 D6 7E 57 55
.:7DC8 FF FF FF FF FF FF AA 59
.:7DD0 FF 70 68 70 9F 6A 7F 6D
.:7DD8 F0 C3 F0 CF C3 F3 F0 CB
.:7DE0 EA 2A 3A 0A 3A C2 3E BA
.:7DE8 FF FF FF FF FF FF FF FF
.:7DF0 FA F5 FA F5 FA F5 FA F5
.:7DF8 FF FF FF FF FF FF FF FF
.:7E00 FA F5 FA F5 FA F5 FA F5
.:7E08 FA AF FA AF FA AF FA AF
.:7E10 FA FF AF FA AF FA AF FA
.:7E18 AA AF FA AF FA AF FA AF
.:7E20 F9 AD F9 AD F9 AD F9 AD
.:7E28 FF FF FF FF FE FA EA AA
.:7E30 FF FA EA AA AA AA AA AA
.:7E38 AA AA AA AA AA AA AA AA
.:7E40 AA AA AA AA AA AA AA AA
.:7E48 AA AF BF FF BF AB AA AA
.:7E50 E5 DA E5 DA E5 DA E5 DA
.:7E58 D5 7D F7 7F 55 7D F7 7F
.:7E60 E9 65 E5 65 EB 65 E5 E9
.:7E68 57 7D F7 55 DD 7D F5 5F
.:7E70 EA AF FA BF FA AF FA AF
.:7E78 5A A5 56 A5 5A A9 56 A9
.:7E80 FA AA BE AA FA AF FA AF
.:7E88 F5 5F F5 5F F5 5F F5 5F
.:7E90 77 5D F5 5F F5 5F F5 5F
.:7E98 55 F5 D7 7D D7 F5 F7 5D
.:7EA0 FA AA EB BE FA AE EB BE
.:7EA8 55 F5 5F F5 5F F5 5F F5
.:7EB0 5F F5 55 7D D7 7D D7 7D
.:7EB8 5F F5 55 7D D7 7D D7 7D
.:7EC0 FA AF FF EB BF EB BE EB
.:7EC8 5F F5 5F F5 5F F5 5F F5
.:7ED0 FA AF FA AF FA AF FA AF
.:7ED8 FF AE FB AF FA AE FB AE
.:7EE0 EB AE FA AF FA AE FB AF
.:7EE8 D5 FF D5 5F 55 FF 5D 5F
.:7EF0 FB BE AA AF FA BE BA AF
.:7EF8 55 7F F7 55 F5 5F F7 55
.:7F00 EB FE BF AB EB FE BF EB
.:7F08 F5 DF 55 7F F5 DF 55 7D
.:7F10 E7 BD FA AF FA AF FA AF
.:7F18 E5 6E A9 6F E5 6F 67 6F
.:7F20 E5 5A A5 5A A5 6A A5 5A
.:7F28 AF FA AF FA AF FA AF EA
.:7F30 FA AF FA AF FA AF FA BF
.:7F38 FF FF FF FF FF FF FF FF
.:7F40 EF FF E7 22 EF 20 FF FB
.:7F48 EF 71 FF 00 FF 15 FF 20
.:7F50 EF 30 FF 7F FF 20 FF 11
.:7F58 FF 38 FF 04 FF 73 7C FF
.:7F60 CF 32 FF 7B FF 00 FF 00
.:7F68 FF 02 FF 10 7B 30 FF FF
.:7F70 FF FB FF 30 FF 00 FF 31
.:7F78 7B 20 14 20 73 31 FF 73
.:7F80 FF 38 EF 30 FF 00 FF FF
.:7F88 EF 10 FF D3 FF 00 DF FB
.:7F90 FF 00 DF 00 FF 00 11 30
.:7F98 FF 00 FB FF FD 0D FD 3D
.:7FA0 FD 0D FD 10 FD FD FD 3D
.:7FA8 FD FD FF FB 7F 00 FF FF
.:7FB0 FF 00 10 02 FF 00 CF 20
.:7FB8 FF 00 4C 10 F3 30 57 00
```

```
.:7FC0 0D 6D D3 D3 0D FD 0D ED
.:7FC8 1D FF 08 FF 20 FD 0D DF
.:7FD0 0D FD D3 D3 0D FD 00 FF
.:7FD8 28 FF 00 FB 00 CF DB FF
.:7FE0 38 FF 00 FF 00 FD 0D D3
.:7FE8 0D FD 0D EF FF FF 01 FC
.:7FF0 20 FF 20 FF 10 FF 00 FF
.:7FF8 FF FF FF DF ED 1D D3 ED
.:8000 4C FC 88 28 43 29 20 43
.:8008 4F 50 59 52 49 47 48 54
.:8010 20 31 39 38 33 2C 20 53
.:8018 54 45 56 45 4E 20 42 52
.:8020 45 53 53 28 50 29 20 50
.:8028 45 52 46 4F 52 4D 41 4E
.:8030 43 45 20 31 39 38 33 2C
.:8038 20 53 54 45 56 45 4E 20
.:8040 42 52 45 53 53 3F C0 C3
.:8048 CC F0 C0 3F 00 00 C0 C0
.:8050 C0 C0 C0 00 00 0C 3C 0C
.:8058 0C 0C 0C 3F 00 00 00 00
.:8060 00 00 00 00 00 3F C0 00
.:8068 0F 30 C0 FF 00 00 C0 C0
.:8070 00 00 00 C0 00 FF 00 03
.:8078 0F 00 C0 3F 00 C0 C0 00
.:8080 00 C0 C0 00 00 03 0F 33
.:8088 C3 FF 03 03 00 00 00 00
.:8090 00 C0 00 00 00 FF C0 FF
.:8098 00 00 C0 3F 00 C0 00 00
.:80A0 C0 C0 C0 00 00 0F 30 C0
.:80A8 FF C0 C0 3F 00 C0 00 00
.:80B0 00 C0 C0 00 00 FF 00 03
.:80B8 0C 30 30 30 00 C0 C0 00
.:80C0 00 00 00 00 00 3F C0 C0
.:80C8 3F C0 C0 3F 00 00 C0 C0
.:80D0 00 C0 C0 00 00 3F C0 C0
.:80D8 3F 00 03 FC 00 00 C0 C0
.:80E0 C0 C0 00 00 00 00 00 00
.:80E8 00 00 00 00 00 00 00 00
.:80F0 00 00 00 00 00 98 90 85
.:80F8 80 75 70 65 60 55 50 45
.:8100 40 35 20 45 30 01 01 01
.:8108 01 01 01 01 00 00 00 00
.:8110 00 00 00 00 00 10 08 14
.:8118 20 35 99 99 99 04 04 05
.:8120 06 09 0A 06 06 05 05 0A
.:8128 09 09 08 01 02 08 08 09
.:8130 0A 05 06 0A 0A 09 09 06
.:8138 05 08 04 02 01 FC F3 3F
.:8140 CF FE FD FB F7 EF DF BF
.:8148 7F 01 02 04 08 10 20 40
.:8150 80 00 03 08 05 08 06 05
.:8158 0A 07 01 01 01 02 01 02
.:8160 03 03 04 40 47 05 47 05
.:8168 47 06 47 05 0C 07 47 05
.:8170 47 06 47 05 47 05 ED 05
.:8178 47 05 47 06 47 05 ED 05
.:8180 47 05 47 05 47 06 47 05
.:8188 0C 07 47 05 47 06 47 05
.:8190 47 05 ED 05 47 05 47 06
.:8198 47 05 ED 05 47 05 47 05
.:81A0 47 06 47 05 0C 07 47 05
.:81A8 47 06 47 05 47 05 ED 05
.:81B0 47 05 47 06 47 05 ED 05
.:81B8 E9 07 0C 07 47 06 0C 07
.:81C0 E9 07 0C 07 47 06 0C 07
.:81C8 47 05 47 05 00 00 1E 0F
.:81D0 0F 0F 0F 0F 0F 1E 0F 0F
.:81D8 0F 0F 0F 0F 1E 0F 0F 0F
.:81E0 0F 0F 0F 1E 0F 0F 0F 0F
.:81E8 0F 0F 1E 0F 0F 0F 0F 0F
.:81F0 0F 1E 0F 0F 0F 0F 0F 2D
.:81F8 3C 3C 3C 3C 3C 3C 3C 3C
.:8200 3C 3C 00 00 A5 1F 31 1C
.:8208 32 19 B5 17 31 1C 32 19
.:8210 B5 17 1F 15 32 19 B5 17
.:8218 1F 15 D1 12 B5 17 1F 15
.:8220 D1 12 D2 0F 1F 15 B5 17
.:8228 32 19 B5 17 D1 12 1F 15
.:8230 B5 17 32 19 B5 17 1F 15
.:8238 32 19 31 1C 32 19 B5 17
.:8240 31 1C A5 1F 31 1C 32 19
.:8248 A5 1F 87 21 A5 1F 31 1C
.:8250 32 19 B5 17 1F 15 32 19
.:8258 B5 17 D1 12 D2 0F B5 17
.:8260 1F 15 A5 1F 32 19 1F 15
.:8268 A5 1F 32 19 1F 15 A5 1F
.:8270 32 19 1F 15 31 1C B5 17
.:8278 D1 12 31 1C B5 17 D1 12
.:8280 31 1C B5 17 D1 12 32 19
.:8288 1F 15 C3 10 32 19 1F 15
.:8290 C3 10 32 19 1F 15 C3 10
.:8298 31 1C B5 17 D1 12 31 1C
.:82A0 B5 17 D1 12 31 1C B5 17
.:82A8 D1 12 A5 1F 32 19 1F 15
.:82B0 A5 1F 32 19 1F 15 A5 1F
.:82B8 32 19 1F 15 31 1C B5 17
.:82C0 D1 12 31 1C B5 17 D1 12
.:82C8 31 1C B5 17 D1 12 32 19
.:82D0 1F 15 C3 10 32 19 1F 15
.:82D8 C3 10 32 19 1F 15 C3 10
.:82E0 31 1C B5 17 D1 12 31 1C
.:82E8 B5 17 D1 12 31 1C B5 17
.:82F0 D1 12 1F 15 B5 17 D1 12
.:82F8 A5 1F 32 19 00 00 0F 0F
.:8300 0F 0F 0F 0F 0F 0F 0F 0F
.:8308 0F 0F 0F 0F 0F 0F 0F 0F
.:8310 0F 0F 0F 2D 0F 0F 0F 0F
.:8318 0F 0F 0F 0F 0F 0F 0F 0F
.:8320 0F 0F 0F 0F 0F 0F 0F 0F
.:8328 0F 0F 0F 0F 1E 07 07 06
.:8330 07 07 06 07 07 06 07 07
.:8338 06 07 07 06 07 07 06 07
.:8340 07 06 07 07 06 07 07 06
.:8348 07 07 06 07 07 06 07 07
.:8350 06 07 07 06 07 07 06 07
.:8358 07 06 07 07 06 07 07 06
.:8360 07 07 06 07 07 06 07 07
.:8368 06 07 07 06 07 07 06 07
```

```
.:8370 07 06 07 07 06 0F 0F 0F
.:8378 0F 3C 00 00 78 A9 06 85
.:8380 01 A9 00 8D 0D DC 8D 0D
.:8388 DD AD 0D DC AD 0D DD A9
.:8390 00 8D 02 DC 8D 03 DC 8D
.:8398 0E DC 8D 0F DC AD 0E DD
.:83A0 AD 0F DD A9 63 8D 18 03
.:83A8 A9 81 8D 19 03 A9 23 8D
.:83B0 14 03 A9 87 8D 15 03 A9
.:83B8 00 85 02 A9 5C 85 03 A9
.:83C0 00 85 04 A9 9C 85 05 A0
.:83C8 00 A2 00 B1 02 91 04 C8
.:83D0 D0 F9 E6 03 E6 05 E8 E0
.:83D8 24 D0 F0 A9 00 85 02 A9
.:83E0 20 85 03 A9 00 85 04 A9
.:83E8 60 85 05 A0 00 A2 00 B1
.:83F0 02 91 04 C8 D0 F9 E6 03
.:83F8 E6 05 E8 E0 20 D0 F0 A9
.:8400 00 85 02 A9 14 85 03 A9
.:8408 00 85 04 A9 5C 85 05 A0
.:8410 00 A2 00 B1 02 91 04 C8
.:8418 D0 F9 E6 03 E6 05 E8 E0
.:8420 04 D0 F0 A9 00 85 02 A9
.:8428 18 85 03 A9 00 85 04 A9
.:8430 D8 85 05 A0 00 A2 00 B1
.:8438 02 91 04 C8 D0 F9 E6 03
.:8440 E6 05 E8 E0 04 D0 F0 A9
.:8448 01 8D 1A D0 A9 FF 8D 19
.:8450 D0 A9 21 85 29 A9 04 85
.:8458 32 A9 82 85 33 A9 FE 85
.:8460 38 A9 82 85 39 A9 19 8D
.:8468 0C D4 A9 95 8D 0D D4 A9
.:8470 00 8D 1C D0 A9 00 85 4F
.:8478 A5 50 85 51 85 52 85 53
.:8480 85 54 85 55 A9 E0 85 56
.:8488 A9 30 85 43 A9 09 8D FF
.:8490 5F A9 0A 8D F8 5F A9 0B
.:8498 8D F9 5F A9 0C 8D FA 5F
.:84A0 A9 0D 8D FB 5F A9 0E 8D
.:84A8 FC 5F A9 0F 8D FD 5F A9
.:84B0 01 85 18 85 19 85 1A 85
.:84B8 1B 85 1C 85 1D 85 1E 85
.:84C0 1F A9 28 85 3C A9 58 85
.:84C8 3D A9 88 85 3E A9 B8 85
.:84D0 3F A9 E8 85 40 A9 18 85
.:84D8 41 A9 3E 85 42 A9 20 85
.:84E0 44 A9 BF 8D 15 D0 A9 7F
.:84E8 8D 1D D0 8D 17 D0 A9 0F
.:84F0 8D 21 D0 A9 00 8D 20 D0
.:84F8 A9 20 4A 4A 85 95 AD 18
.:8500 D0 29 F0 05 95 8D 18 D0
.:8508 AD 11 D0 09 20 8D 11 D0
.:8510 AD 16 D0 09 10 8D 16 D0
.:8518 A2 00 A9 1C 0A 0A 85 95
.:8520 AD 18 D0 29 0F 05 95 8D
.:8528 18 D0 A9 03 8D 02 DD 38
.:8530 E9 01 8D 00 DD A9 64 85
.:8538 30 A9 81 85 31 A9 CE 85
.:8540 36 A9 81 85 37 A9 25 8D
.:8548 05 D4 A9 84 8D 06 D4 A9
.:8550 0F 8D 18 D4 A0 00 B1 30
.:8558 8D 00 D4 B1 32 8D 07 D4
.:8560 C8 B1 30 8D 01 D4 B1 32
.:8568 8D 08 D4 A9 20 8D 04 D4
.:8570 8D 0B D4 A9 21 8D 04 D4
.:8578 8D 0B D4 85 28 85 29 A5
.:8580 30 18 69 02 85 30 A9 00
.:8588 65 31 85 31 A5 32 18 69
.:8590 02 85 32 A9 00 65 33 85
.:8598 33 A0 00 B1 36 85 6A B1
.:85A0 38 85 6B A5 38 18 69 01
.:85A8 85 38 A9 00 65 39 85 39
.:85B0 A5 36 18 69 01 85 36 A9
.:85B8 00 65 37 85 37 A9 FB 8D
.:85C0 12 D0 AD 11 D0 29 7F 8D
.:85C8 11 D0 58 A9 00 85 9D 85
.:85D0 95 8D FE 0F AD 00 DC 2D
.:85D8 01 DC 85 8D 29 10 F0 0E
.:85E0 A5 9D C9 20 90 FE A9 01
.:85E8 8D FE 0F 4C 23 86 AD 1F
.:85F0 D0 85 8E 29 80 F0 DD A2
.:85F8 07 A5 44 3D 49 81 F0 02
.:8600 A9 01 85 9C 66 9C A5 43
.:8608 6A 85 9C A5 9C C9 7A B0
.:8610 07 C9 25 B0 5A 4C C9 86
.:8618 A9 02 8D FF 0F A5 8D 29
.:8620 10 D0 B1 78 A9 00 85 02
.:8628 A9 9C 85 03 A9 00 85 04
.:8630 A9 5C 85 05 A0 00 A2 00
.:8638 B1 02 91 04 C8 D0 F9 E6
.:8640 03 E6 05 E8 E0 24 D0 F0
.:8648 A9 00 85 02 A9 10 85 03
.:8650 A9 00 85 04 A9 D8 85 05
.:8658 A0 00 A2 00 B1 02 91 04
.:8660 C8 D0 F9 E6 03 E6 05 E8
.:8668 E0 04 D0 F0 4C 33 89 A9
.:8670 01 8D FF 0F A5 8D 29 10
.:8678 F0 03 4C D4 85 78 A9 00
.:8680 85 02 A9 9C 85 03 A9 00
.:8688 85 04 A9 5C 85 05 A0 00
.:8690 A2 00 B1 02 91 04 C8 D0
.:8698 F9 E6 03 E6 05 E8 E0 24
.:86A0 D0 F0 A9 00 85 02 A9 10
.:86A8 85 03 A9 00 85 04 A9 D8
.:86B0 85 05 A0 00 A2 00 B1 02
.:86B8 91 04 C8 D0 F9 E6 03 E6
.:86C0 05 E8 E0 04 D0 F0 4C 33
.:86C8 89 A9 00 8D FF 0F A5 8D
.:86D0 29 10 F0 03 4C D4 85 78
.:86D8 A9 00 85 02 A9 9C 85 03
.:86E0 A9 00 85 04 A9 5C 85 05
.:86E8 A0 00 A2 00 B1 02 91 04
.:86F0 C8 D0 F9 E6 03 E6 05 E8
.:86F8 E0 24 D0 F0 A9 00 85 02
.:8700 A9 10 85 03 A9 00 85 04
.:8708 A9 D8 85 05 A0 00 A2 00
.:8710 B1 02 91 04 C8 D0 F9 E6
.:8718 03 E6 05 E8 E0 04 D0 F0
```

```
.:8720 4C 33 89 78 E6 9F E6 9E
.:8728 A5 9E 49 3C D0 04 85 9E
.:8730 E6 9D A5 9F 29 03 D0 27
.:8738 A5 44 8D 10 D0 A5 4F C9
.:8740 5C B0 1C A2 00 A0 00 B5
.:8748 3C 99 00 D0 B5 4F 99 01
.:8750 D0 B5 18 9D 27 D0 F6 4F
.:8758 E8 C8 C8 E0 07 D0 E8 A5
.:8760 56 8D 0F D0 A5 43 8D 0E
.:8768 D0 A5 1F 8D 2E D0 A5 95
.:8770 F0 03 4C 39 88 C6 6A A5
.:8778 6A D0 5D A0 00 B1 36 D0
.:8780 1E A9 64 85 30 A9 81 85
.:8788 31 A9 CE 85 36 A9 81 85
.:8790 37 A9 01 85 95 A9 40 8D
.:8798 04 D4 8D 0B D4 B1 36 85
.:87A0 6A B1 30 8D 00 D4 C8 B1
.:87A8 30 8D 01 D4 A5 30 18 69
.:87B0 02 85 30 A9 00 65 31 85
.:87B8 31 A5 36 18 69 01 85 36
.:87C0 A9 00 65 37 85 37 A5 95
.:87C8 D0 0E A5 28 29 FE 8D 04
.:87D0 D4 09 01 8D 04 D4 85 28
.:87D8 C6 6B D0 5D A0 00 B1 38
.:87E0 D0 22 A9 04 85 32 A9 82
.:87E8 85 33 A9 FE 85 38 A9 82
.:87F0 85 39 A9 01 85 95 A9 40
.:87F8 8D 04 D4 8D 0B D4 B1 38
.:8800 A5 95 D0 35 85 6B B1 32
.:8808 8D 07 D4 C8 B1 32 8D 08
.:8810 D4 A5 32 18 69 02 85 32
.:8818 A9 00 65 33 85 33 A5 38
.:8820 18 69 01 85 38 A9 00 65
.:8828 39 85 39 A5 29 29 FE 8D
.:8830 0B D4 09 01 8D 0B D4 85
.:8838 29 A5 8D 29 08 D0 2E A2
.:8840 07 B5 3C 18 69 01 95 3C
.:8848 90 07 A5 44 1D 49 81 85
.:8850 44 A5 43 C9 3F D0 16 A5
.:8858 44 29 80 F0 10 B5 3C 38
.:8860 E9 01 95 3C B0 07 A5 44
.:8868 3D 41 81 85 44 A5 8D 29
.:8870 04 D0 2E A2 07 B5 3C 38
.:8878 E9 01 95 3C B0 07 A5 44
.:8880 3D 41 81 85 44 A5 43 C9
.:8888 18 D0 16 A5 44 29 80 D0
.:8890 10 B5 3C 18 69 01 95 3C
.:8898 90 07 A5 44 1D 49 81 85
.:88A0 44 A9 FB 8D 12 D0 AD 11
.:88A8 D0 29 7F 8D 11 D0 A9 FF
.:88B0 8D 19 D0 68 A8 68 AA 68
.:88B8 40 EA EA 03 0E 05 07 08
.:88C0 0A 02 02 03 00 01 20 21
.:88C8 03 0D 07 08 0A 02 03 0D
.:88D0 07 08 0A 02 03 0D 07 08
.:88D8 06 05 05 01 01 01 00 01
.:88E0 00 00 00 00 00 00 00 00
.:88E8 00 00 00 04 00 00 03 01
.:88F0 0A 08 07 0D 0E 03 01 0A
.:88F8 08 07 0D 0E EA 78 A9 00
.:8900 8D 0D DC 8D 0D DD AD 0D
.:8908 DC AD 0D DD A9 00 8D 1B
.:8910 D0 8D 02 DC 8D 03 DC 8D
.:8918 0E DC 8D 0F DC 8D 0E DD
.:8920 8D 0F DD A2 FF 9A A9 63
.:8928 8D 18 03 A9 81 8D 19 03
.:8930 4C 7C 83 A9 0F 8D 14 03
.:8938 A9 95 8D 15 03 A2 02 A9
.:8940 00 A8 95 00 E8 D0 FB 85
.:8948 10 A9 A5 8D 05 D4 8D 0C
.:8950 D4 A9 00 8D 06 D4 A9 04
.:8958 8D 0D D4 A9 80 8D 16 D4
.:8960 85 28 8D 00 D4 8D 07 D4
.:8968 A9 01 8D 01 D4 8D 16 D4
.:8970 A9 20 8D 08 D4 85 29 8D
.:8978 0B D4 A9 9F 8D 18 D4 A9
.:8980 01 85 8C 8D 1A D0 A9 FF
.:8988 8D 1C D0 8D 19 D0 A9 45
.:8990 A9 C0 85 7D 85 7E A9 03
.:8998 85 82 A9 0F 85 81 A9 90
.:89A0 85 55 A9 EB 85 56 A9 FF
.:89A8 8D 15 D0 A9 8B 8D 12 D4
.:89B0 A9 80 8D 12 D4 85 2A A9
.:89B8 FE 8D 0E D4 8D 0F D4 A9
.:89C0 0F 85 83 A9 40 85 84 85
.:89C8 85 85 86 85 87 A9 3B 8D
.:89D0 11 D0 A9 78 8D 18 D0 A9
.:89D8 00 85 76 A9 18 8D 16 D0
.:89E0 A9 00 8D 1D D0 8D 17 D0
.:89E8 A2 00 A9 00 85 6C 85 6D
.:89F0 85 70 85 6F A8 85 02 85
.:89F8 04 A9 D8 85 05 A9 10 85
.:8A00 03 B1 02 91 04 C8 D0 F9
.:8A08 E6 03 A6 05 E8 86 05 E0
.:8A10 DC D0 EE A2 00 A0 01 98
.:8A18 95 18 E8 C8 E0 08 D0 F7
.:8A20 A0 00 A2 09 98 99 F8 5F
.:8A28 99 20 00 E8 C8 98 C0 08
.:8A30 D0 F3 A0 06 A9 04 99 20
.:8A38 00 A0 07 A9 05 99 20 00
.:8A40 A9 06 85 24 A9 07 85 25
.:8A48 85 26 A9 00 85 A9 A9 14
.:8A50 85 AA A9 00 85 A7 A9 18
.:8A58 85 A8 A9 00 85 AB A9 1C
.:8A60 85 AC A9 60 85 3F A9 07
.:8A68 8D 25 D0 A9 03 85 18 85
.:8A70 19 85 1F 8D 26 D0 A9 01
.:8A78 85 1A 85 1B 85 1C 85 1D
.:8A80 85 1E A9 07 8D 2E D0 AD
.:8A88 FF 0F 85 65 A2 00 A0 00
.:8A90 A9 00 85 8D A9 1C 85 8E
.:8A98 B1 8D 85 95 10 2E C9 FF
.:8AA0 D0 03 4C 1E 8B 29 0F 85
.:8AA8 95 C8 B1 8D 85 8F A9 D8
.:8AB0 18 C8 71 8D 85 90 A5 8D
.:8AB8 18 69 03 85 8D A9 00 65
.:8AC0 8E 85 8E A9 04 A0 00 91
.:8AC8 8F 4C 98 8A 29 40 F0 27
```

```
.:8AD0 C8 B1 8D 85 8F A9 5C 18
.:8AD8 C8 71 8D 85 90 A5 8D 18
.:8AE0 69 03 85 8D A9 00 65 8E
.:8AE8 85 8E A0 00 B1 8F 29 F0
.:8AF0 09 04 91 8F 4C 98 8A C8
.:8AF8 B1 8D 85 8F A9 5C 18 C8
.:8B00 71 8D 85 90 A5 8D 18 69
.:8B08 03 85 8D A9 00 65 8E 85
.:8B10 8E A0 00 B1 8F 29 0F 09
.:8B18 40 91 8F 4C 98 8A A9 03
.:8B20 8D 02 DD AD 00 DD 29 FC
.:8B28 09 02 8D 00 DD A9 00 8D
.:8B30 20 D0 A9 FB 8D 12 D0 AD
.:8B38 11 D0 29 7F 8D 11 D0 58
.:8B40 A9 00 85 A0 AD FE 0F F0
.:8B48 0D AD 00 DC 2D 01 DC 29
.:8B50 10 D0 03 4C 7C 83 A6 4E
.:8B58 BD 5A 81 85 9C A2 00 A5
.:8B60 8B C9 06 D0 03 4C E8 8B
.:8B68 A5 83 3D 49 81 D0 79 A5
.:8B70 10 3D 49 81 D0 72 A5 9F
.:8B78 29 3F DD 15 81 F0 05 B5
.:8B80 77 4C 96 8B A5 A1 29 0F
.:8B88 A8 B5 72 D0 06 B9 2D 81
.:8B90 4C 96 8B B9 1D 81 95 77
.:8B98 85 8D 66 8D 90 10 B5 3C
.:8BA0 18 65 9C 95 3C 90 07 A5
.:8BA8 44 1D 49 81 85 44 66 8D
.:8BB0 90 10 B5 3C 38 E5 9C 95
.:8BB8 3C B0 07 A5 44 3D 41 81
.:8BC0 85 44 66 8D 90 0F B5 4F
.:8BC8 38 E5 9C 95 4F C9 40 B0
.:8BD0 04 A9 40 95 4F 66 8D 90
.:8BD8 0F B5 4F 18 65 9C 95 4F
.:8BE0 C9 EB 90 04 A9 EB 95 4F
.:8BE8 E8 E0 04 F0 03 4C 5F 8B
.:8BF0 A2 00 B5 3C C9 18 B0 12
.:8BF8 A5 44 3D 49 81 D0 0B B5
.:8C00 77 29 FD 09 01 95 77 4C
.:8C08 1F 8C B5 3C C9 3F 90 0F
.:8C10 A5 44 3D 49 81 F0 08 B5
.:8C18 77 29 FE 09 02 95 77 E8
.:8C20 E0 04 D0 CE A5 8B C9 06
.:8C28 D0 09 A5 9D C9 30 90 03
.:8C30 4C 7C 83 A5 82 29 01 D0
.:8C38 41 A5 8B C9 06 F0 3B A5
.:8C40 7D D0 37 A5 65 C9 02 F0
.:8C48 31 AD 00 DC 09 EF C9 FF
.:8C50 F0 28 A5 82 09 01 85 82
.:8C58 A5 55 85 5E A2 06 A5 44
.:8C60 3D 49 81 F0 02 A9 01 85
.:8C68 8E A5 42 85 8D 66 8E A5
.:8C70 8D 6A 18 69 08 85 4C 20
.:8C78 00 97 A5 82 29 02 D0 41
.:8C80 A5 8B C9 06 F0 3B A5 7E
.:8C88 D0 37 A5 65 C9 01 F0 31
.:8C90 AD 01 DC 09 EF C9 FF F0
.:8C98 28 A5 82 09 02 85 82 A9
.:8CA0 F0 85 5D A2 07 A5 44 3D
.:8CA8 49 81 F0 02 A9 01 85 8E
.:8CB0 A5 43 85 8D 66 8E A5 8D
.:8CB8 6A 18 69 08 85 4B 20 2F
.:8CC0 97 A5 65 C9 02 D0 07 A9
.:8CC8 05 85 55 4C 84 8D AD 00
.:8CD0 DC 09 F0 A8 A5 7D F0 03
.:8CD8 4C 84 8D A5 8B C9 06 D0
.:8CE0 03 4C 84 8D 98 09 FE C9
.:8CE8 FF F0 0A C6 55 A5 55 C9
.:8CF0 48 D0 02 E6 55 98 09 FD
.:8CF8 C9 FF F0 0A E6 55 A5 55
.:8D00 C9 D5 D0 02 C6 55 98 09
.:8D08 FB C9 FF F0 38 A2 06 B5
.:8D10 3C 38 E9 01 95 3C B0 07
.:8D18 A5 44 3D 41 81 85 44 B5
.:8D20 3C 38 E9 01 95 3C B0 07
.:8D28 A5 44 3D 41 81 85 44 A5
.:8D30 4E C9 04 90 10 B5 3C 38
.:8D38 E9 01 95 3C B0 07 A5 44
.:8D40 3D 41 81 85 44 98 09 F7
.:8D48 C9 FF F0 38 A2 06 B5 3C
.:8D50 18 69 01 95 3C 90 07 A5
.:8D58 44 1D 49 81 85 44 B5 3C
.:8D60 18 69 01 95 3C 90 07 A5
.:8D68 44 1D 49 81 85 44 A5 4E
.:8D70 C9 04 90 10 B5 3C 18 69
.:8D78 01 95 3C 90 07 A5 44 1D
.:8D80 49 81 85 44 AD 01 DC A8
.:8D88 A5 7E F0 03 4C 23 8E A5
.:8D90 8B C9 06 D0 03 4C 23 8E
.:8D98 A5 65 C9 01 D0 07 A9 05
.:8DA0 85 56 4C 23 8E 98 09 FB
.:8DA8 C9 FF F0 38 A2 07 B5 3C
.:8DB0 38 E9 01 95 3C B0 07 A5
.:8DB8 44 3D 41 81 85 44 B5 3C
.:8DC0 38 E9 01 95 3C B0 07 A5
.:8DC8 44 3D 41 81 85 44 A5 4E
.:8DD0 C9 04 90 10 B5 3C 38 E9
.:8DD8 01 95 3C B0 07 A5 44 3D
.:8DE0 41 81 85 44 98 09 F7 C9
.:8DE8 FF F0 38 A2 07 B5 3C 18
.:8DF0 69 01 95 3C 90 07 A5 44
.:8DF8 1D 49 81 85 44 B5 3C 18
.:8E00 69 01 95 3C 90 07 A5 44
.:8E08 1D 49 81 85 44 A5 4E C9
.:8E10 04 90 10 B5 3C 18 69 01
.:8E18 95 3C 90 07 A5 44 1D 49
.:8E20 81 85 44 A2 00 B5 4F C9
.:8E28 E8 90 0B B5 72 D0 07 A9
.:8E30 01 95 72 20 5E 97 E8 E0
.:8E38 04 D0 EA A2 00 86 8D B5
.:8E40 4F C9 43 B0 14 B5 72 F0
.:8E48 10 E6 76 E6 76 E6 76 E6
.:8E50 76 E6 76 A9 00 95 72 E6
.:8E58 8D E8 E0 04 D0 E1 A5 8D
.:8E60 F0 36 A2 00 A5 67 C9 40
.:8E68 D0 02 A9 3F 85 67 4A 4A
.:8E70 A8 F8 A5 8D F0 22 B9 F5
.:8E78 80 18 65 5F 85 5F A9 00
```

```
.:8E80 65 60 85 60 A9 00 85 67
.:8E88 65 61 85 61 20 8D 97 C6
.:8E90 8D F0 05 E8 E0 04 D0 D9
.:8E98 D8 A2 00 B5 42 C9 18 B0
.:8EA0 0C A5 44 3D 4F 81 D0 05
.:8EA8 F6 42 4C 9B 8E B5 42 C9
.:8EB0 3F 90 0C A5 44 3D 4F 81
.:8EB8 F0 05 D6 42 4C AD 8E E8
.:8EC0 E0 02 D0 D7 A5 8B C9 06
.:8EC8 F0 58 A5 9F 29 03 D0 4D
.:8ED0 A2 00 A5 83 85 8E A5 81
.:8ED8 85 8D 66 8E 90 05 66 8D
.:8EE0 4C 1D 8F 66 8D B0 36 A5
.:8EE8 A2 A4 4E D9 51 81 B0 2D
.:8EF0 A5 44 3D 49 81 F0 02 A9
.:8EF8 01 85 96 B5 3C 85 95 66
.:8F00 96 A5 95 6A 18 69 06 95
.:8F08 45 B5 4F 18 69 0A 95 57
.:8F10 A5 81 1D 49 81 85 81 20
.:8F18 BC 97 4C 22 8F E8 E0 04
.:8F20 D0 B8 A5 9F 29 20 F0 05
.:8F28 A0 01 4C 2F 8F A0 00 A5
.:8F30 83 85 8D A2 00 66 8D B0
.:8F38 14 B5 72 F0 08 B9 C2 88
.:8F40 95 20 4C 52 8F B9 C4 88
.:8F48 95 20 4C 52 8F B9 C6 88
.:8F50 95 20 E8 E0 04 D0 DE A5
.:8F58 76 D0 03 4C 13 90 A5 9F
.:8F60 29 07 C9 06 F0 03 4C 13
.:8F68 90 A2 00 A0 00 B1 AB C9
.:8F70 FF D0 07 A9 06 85 8B 4C
.:8F78 13 90 E6 0F 85 95 A5 95
.:8F80 10 29 C8 B1 AB 85 8D C8
.:8F88 A9 D8 18 71 AB 85 8E A5
.:8F90 95 A0 00 29 0F 91 8D A5
.:8F98 AB 18 69 03 85 AB A9 00
.:8FA0 65 AC 85 AC C6 76 D0 C1
.:8FA8 4C 13 90 29 40 F0 31 C8
.:8FB0 B1 AB 85 8D C8 A9 5C 18
.:8FB8 71 AB 85 8E A5 95 29 0F
.:8FC0 85 95 A0 00 B1 8D 29 F0
.:8FC8 05 95 91 8D A5 AB 18 69
.:8FD0 03 85 AB A9 00 65 AC 85
.:8FD8 AC C6 76 D0 8C 4C 13 90
.:8FE0 C8 B1 AB 85 8D C8 A9 5C
.:8FE8 18 71 AB 85 8E A5 95 0A
.:8FF0 0A 0A 0A 85 95 A0 00 B1
.:8FF8 8D 29 0F 05 95 91 8D A5
.:9000 AB 18 69 03 85 AB A9 00
.:9008 65 AC 85 AC C6 76 F0 03
.:9010 4C 69 8F A5 8B C9 06 D0
.:9018 12 20 78 98 20 7F 98 A5
.:9020 66 D0 08 A9 00 85 9D A9
.:9028 01 85 66 A9 00 85 8A A2
.:9030 07 A5 44 3D 49 81 F0 02
.:9038 A9 01 85 8E 66 8E A5 43
.:9040 6A 85 8E 18 69 09 85 8E
.:9048 CA A5 44 3D 49 81 F0 02
.:9050 A9 01 85 8D 66 8D A5 42
.:9058 6A 85 8D 18 69 09 85 8D
.:9060 A0 00 A2 00 A5 81 85 96
.:9068 66 96 90 1D B9 8D 00 38
.:9070 F5 45 C9 0E B0 13 B9 55
.:9078 00 69 0C 38 F5 57 C9 0C
.:9080 B0 07 A5 8A 19 4F 81 85
.:9088 8A E8 E0 04 D0 DA A2 00
.:9090 A5 81 85 96 C8 C0 02 D0
.:9098 CF A2 03 A5 44 3D 49 81
.:90A0 F0 02 A9 01 85 90 66 90
.:90A8 A5 3F 6A 85 90 CA A5 44
.:90B0 3D 49 81 F0 02 A9 01 85
.:90B8 8F 66 8F A5 3E 6A 85 8F
.:90C0 CA A5 44 3D 49 81 F0 02
.:90C8 A9 01 85 8E 66 8E A5 3D
.:90D0 6A 85 8E CA A5 44 3D 49
.:90D8 81 F0 02 A9 01 85 8D 66
.:90E0 8D A5 3C 6A 85 8D A0 00
.:90E8 A2 00 B9 8D 00 18 69 0B
.:90F0 38 F5 4B C9 0E B0 36 B9
.:90F8 4F 00 69 0A 38 F5 5D C9
.:9100 0A B0 2A A5 8A 19 49 81
.:9108 85 8A F8 8A D0 11 A5 6C
.:9110 18 69 01 85 6C A9 00 65
.:9118 6D 85 6D D8 4C 2D 91 A5
.:9120 6F 18 69 01 85 6F A9 00
.:9128 65 70 85 70 D8 C8 C0 04
.:9130 D0 B8 A0 00 E8 E0 02 D0
.:9138 B1 A2 00 A5 8A 29 C0 F0
.:9140 3B 24 8A 10 0C A5 7E D0
.:9148 08 A9 80 85 7E 20 EB 97
.:9150 E8 24 8A 50 0C A5 7D D0
.:9158 08 A9 80 85 7D 20 1A 98
.:9160 E8 8A F0 18 F8 A9 25 18
.:9168 65 5F 85 5F A9 03 65 60
.:9170 85 60 A9 00 65 61 85 61
.:9178 CA D0 EA D8 A2 00 A5 8A
.:9180 85 8D 29 0F F0 38 66 8D
.:9188 90 2F B5 84 D0 2B A5 83
.:9190 1D 49 81 85 83 A9 80 95
.:9198 84 A9 00 B4 72 95 72 B9
.:91A0 19 81 F8 18 65 62 85 62
.:91A8 A9 00 65 63 85 63 A9 00
.:91B0 65 64 85 64 E6 2D 20 49
.:91B8 98 E8 E0 04 D0 C8 D8 A5
.:91C0 68 D0 49 A5 A6 C9 06 D0
.:91C8 43 A6 65 A5 2D DD D8 88
.:91D0 90 0F A5 4E C9 08 F0 2E
.:91D8 E6 4E A9 09 85 68 4C 06
.:91E0 92 A5 7F C9 14 90 1F A5
.:91E8 2D DD DB 88 B0 18 A5 4E
.:91F0 C9 02 F0 12 C9 01 F0 0E
.:91F8 C9 00 F0 0A C6 4E A9 09
.:9200 85 68 A9 00 85 7F A9 00
.:9208 85 A6 85 2D A9 00 85 8A
.:9210 A5 9F 29 01 F0 27 A5 5F
.:9218 29 0F 85 97 A5 5F 29 F0
.:9220 85 98 A5 60 29 0F 85 99
.:9228 A5 60 29 F0 85 9A A5 61
```

```
.:9230 29 0F 85 9B A5 61 29 F0
.:9238 85 9C 4C 61 92 A5 62 29
.:9240 0F 85 97 A5 62 29 F0 85
.:9248 98 A5 63 29 0F 85 99 A5
.:9250 63 29 F0 85 9A A5 64 29
.:9258 0F 85 9B A5 64 29 F0 85
.:9260 9C A5 97 0A 0A 0A 0A 18
.:9268 69 45 85 8D A5 98 18 69
.:9270 45 85 8F A5 99 0A 0A 0A
.:9278 0A 18 69 45 85 91 A5 9A
.:9280 18 69 45 85 93 A5 9B 0A
.:9288 0A 0A 0A 18 69 45 85 95
.:9290 A5 9C 18 69 45 85 97 A9
.:9298 80 85 8E 85 90 85 92 85
.:92A0 94 85 96 85 98 A9 45 A2
.:92A8 E5 C5 97 D0 1A 86 97 C5
.:92B0 95 D0 14 86 95 C5 93 D0
.:92B8 0E 86 93 C5 91 D0 08 86
.:92C0 91 C5 8F D0 02 86 8F A0
.:92C8 00 A5 9F 29 01 F0 31 A9
.:92D0 D0 85 04 A9 60 85 05 A2
.:92D8 0A B5 8D 85 02 B5 8E 85
.:92E0 03 B1 02 91 04 C8 C0 10
.:92E8 D0 F7 A0 00 A5 04 18 69
.:92F0 10 85 04 A9 00 65 05 85
.:92F8 05 CA CA 10 DC 4C 28 93
.:9300 A9 00 85 04 A9 60 85 05
.:9308 A2 0A B5 8D 85 02 B5 8E
.:9310 85 03 B1 02 91 04 C8 C0
.:9318 10 D0 F7 A0 00 A5 04 18
.:9320 69 10 85 04 CA CA 10 E2
.:9328 A5 81 85 8D A2 00 66 8D
.:9330 90 12 B5 57 18 69 02 95
.:9338 57 C9 FB 90 07 A5 81 3D
.:9340 41 81 85 81 E8 E0 04 D0
.:9348 E5 A5 5D C9 40 90 08 38
.:9350 E9 04 85 5D 4C 61 93 A5
.:9358 82 29 FD 85 82 A9 00 85
.:9360 5D A5 5E C9 40 90 08 38
.:9368 E9 04 85 5E 4C 79 93 A5
.:9370 82 29 FE 85 82 A9 00 85
.:9378 5E EA A5 9F 29 07 C9 06
.:9380 D0 09 A5 0F 4A 85 AD 69
.:9388 00 85 AE A5 9F 29 07 C9
.:9390 06 D0 13 A5 0F 85 9C A5
.:9398 AD 85 0F A9 00 85 AF A9
.:93A0 1C 85 B0 4C BC 93 A5 9F
.:93A8 29 07 C9 07 F0 03 4C AA
.:93B0 94 A5 0F 85 9C A5 AE 85
.:93B8 0F 4C C8 93 E6 2E A5 2E
.:93C0 C9 07 D0 04 A9 00 85 2E
.:93C8 A2 00 E4 0F D0 03 4C AA
.:93D0 94 A0 00 B1 AF 10 3F 86
.:93D8 96 29 0F AA BD DE 88 18
.:93E0 65 2E AA BD EE 88 85 95
.:93E8 A6 96 C8 B1 AF 85 8F C8
.:93F0 A9 D8 18 71 AF 85 90 A0
.:93F8 00 A5 95 91 8F A5 AF 18
.:9400 69 03 85 AF A9 00 65 B0
.:9408 85 B0 E8 E4 0F D0 C4 A5
.:9410 9C 85 0F 4C AA 94 85 97
.:9418 29 40 F0 48 86 96 C8 B1
.:9420 AF 85 8F C8 A9 5C 18 71
.:9428 AF 85 90 A0 00 B1 8F 29
.:9430 F0 85 95 A5 97 29 0F AA
.:9438 BD DE 88 18 65 2E AA BD
.:9440 EE 88 A6 96 05 95 91 8F
.:9448 A5 AF 18 69 03 85 AF A9
.:9450 00 65 B0 85 B0 E8 E4 0F
.:9458 F0 03 4C D3 93 A5 9C 85
.:9460 0F 4C AA 94 86 96 C8 B1
.:9468 AF 85 8F C8 A9 5C 18 71
.:9470 AF 85 90 A0 00 B1 8F 29
.:9478 0F 85 95 A6 97 BD DE 88
.:9480 18 65 2E AA BD EE 88 0A
.:9488 0A 0A 0A 05 95 A6 96 91
.:9490 8F A5 AF 18 69 03 85 AF
.:9498 A9 00 65 B0 85 B0 E8 E4
.:94A0 0F F0 03 4C D3 93 A5 9C
.:94A8 85 0F A5 8C D0 31 A5 83
.:94B0 29 0F C9 0F D0 29 F8 A9
.:94B8 10 18 65 63 85 63 A5 64
.:94C0 69 00 85 64 D8 A5 4E C9
.:94C8 08 F0 02 E6 4E A9 14 85
.:94D0 68 A9 40 85 84 85 85 85
.:94D8 86 85 87 A9 01 85 8C EA
.:94E0 A6 4E E8 8A 0A 0A 0A 0A
.:94E8 A2 00 A8 B9 45 80 9D 98
.:94F0 60 E8 C8 E0 10 D0 F4 A5
.:94F8 8B C9 06 D0 03 4C 8D 98
.:9500 A5 83 29 0F D0 02 85 8C
.:9508 A5 A0 F0 FC 4C 40 8B 78
.:9510 A9 04 8D 21 D0 E6 9F E6
.:9518 9E A9 3C 45 9E D0 18 85
.:9520 9E E6 9D E6 A6 E6 67 E6
.:9528 7F A5 68 F0 0A C6 68 D0
.:9530 06 A9 00 85 A6 85 2D A9
.:9538 01 85 A0 A0 00 A5 28 8D
.:9540 04 D4 A5 29 8D 0B D4 A5
.:9548 2A 8D 12 D4 A5 44 85 2F
.:9550 29 0F 85 2F A5 55 8D 0F
.:9558 D0 A5 42 8D 0E D0 A5 44
.:9560 29 40 F0 06 A5 2F 09 80
.:9568 85 2F A5 9F 29 01 F0 30
.:9570 A5 5D 85 53 A5 4B 0A 85
.:9578 40 90 06 A5 2F 09 10 85
.:9580 2F A5 57 85 54 A5 45 0A
.:9588 85 41 90 06 A5 2F 09 20
.:9590 85 2F A5 46 0A 90 36 A5
.:9598 2F 09 40 85 2F 4C CD 95
.:95A0 A5 5E 85 53 A5 4C 0A 85
.:95A8 40 90 06 A5 2F 09 10 85
.:95B0 2F A5 59 85 54 A5 47 0A
.:95B8 85 41 90 06 A5 2F 09 20
.:95C0 85 2F A5 48 0A 90 06 A5
.:95C8 2F 09 40 85 2F A2 00 B5
.:95D0 84 F0 26 D6 84 B5 84 D0
.:95D8 20 A5 A3 4A 4A 69 40 95
```

```
.:95E0 4F 45 9F 4A 69 18 95 3C
.:95E8 A5 83 3D 41 81 85 83 A5
.:95F0 44 3D 41 81 85 44 4C FE
.:95F8 95 E8 E0 04 D0 D1 A5 7D
.:9600 F0 02 C6 7D A5 7E F0 02
.:9608 C6 7E EA A5 69 F0 07 C6
.:9610 69 D0 03 20 78 98 A5 6A
.:9618 F0 07 C6 6A D0 03 20 7F
.:9620 98 EA A9 B8 8D 14 03 A9
.:9628 96 8D 15 03 A9 E4 8D 12
.:9630 D0 AD 11 D0 29 7F 8D 11
.:9638 D0 A2 00 A0 00 B9 3C 00
.:9640 9D 00 D0 B9 4F 00 9D 01
.:9648 D0 B9 18 00 99 27 D0 B9
.:9650 20 00 99 F8 5F E8 E8 C8
.:9658 C0 08 D0 E1 A9 05 8D 25
.:9660 D0 A9 00 8D 26 D0 A5 2F
.:9668 8D 10 D0 A5 55 8D 0F D0
.:9670 A5 42 8D 0E D0 A5 9F 29
.:9678 01 F0 0E A5 46 0A 8D 0C
.:9680 D0 A5 58 8D 0D D0 4C 94
.:9688 96 A5 48 0A 8D 0C D0 A5
.:9690 5A 8D 0D D0 A2 00 B5 A1
.:9698 6D 1B D4 65 9F 65 9D 75
.:96A0 A2 6A 6A 6A 65 4F 95 A1
.:96A8 E8 E0 04 D0 E9 A9 FF 8D
.:96B0 19 D0 68 A8 68 AA 68 40
.:96B8 78 A9 0D 8D 21 D0 A5 56
.:96C0 8D 0F D0 A5 43 8D 0E D0
.:96C8 AD 10 D0 29 7F 8D 10 D0
.:96D0 A5 44 29 80 F0 08 AD 10
.:96D8 D0 09 80 8D 10 D0 A9 FB
.:96E0 8D 12 D0 AD 11 D0 29 7F
.:96E8 8D 11 D0 A9 0F 8D 14 03
.:96F0 A9 95 8D 15 03 A9 FF 8D
.:96F8 19 D0 68 A8 68 AA 68 40
.:9700 A9 A5 8D 05 D4 A9 85 8D
.:9708 06 D4 A9 03 8D 01 D4 A9
.:9710 20 8D 00 D4 A5 28 29 07
.:9718 09 80 85 28 A5 28 29 FE
.:9720 8D 04 D4 09 01 85 28 8D
.:9728 04 D4 A9 50 85 69 60 A9
.:9730 A5 8D 05 D4 A9 85 8D 06
.:9738 D4 A9 04 8D 01 D4 A9 20
.:9740 8D 00 D4 A5 28 29 07 09
.:9748 80 85 28 A5 28 29 FE 8D
.:9750 04 D4 09 01 85 28 8D 04
.:9758 D4 A9 50 85 69 60 A9 5A
.:9760 8D 05 D4 A9 85 8D 06 D4
.:9768 A9 05 8D 01 D4 A9 20 8D
.:9770 00 D4 A5 28 29 07 09 20
.:9778 85 28 A5 28 29 FE 8D 04
.:9780 D4 09 01 85 28 8D 04 D4
.:9788 A9 20 85 6A 60 A9 5A 8D
.:9790 05 D4 A9 85 8D 06 D4 A9
.:9798 07 8D 01 D4 A9 20 8D 00
.:97A0 D4 A5 28 29 07 09 20 85
.:97A8 28 A5 28 29 FE 8D 04 D4
.:97B0 09 01 85 28 8D 04 D4 A9
.:97B8 20 85 6A 60 A9 77 8D 05
.:97C0 D4 A9 85 8D 06 D4 A9 08
.:97C8 8D 01 D4 A9 30 8D 00 D4
.:97D0 A5 28 29 07 09 80 85 28
.:97D8 A5 28 29 FE 8D 04 D4 09
.:97E0 01 85 28 8D 04 D4 A9 60
.:97E8 85 6A 60 A9 29 8D 05 D4
.:97F0 A9 A5 8D 06 D4 A9 04 8D
.:97F8 01 D4 A9 30 8D 00 D4 A5
.:9800 28 29 07 09 20 85 28 A5
.:9808 28 29 FE 8D 04 D4 09 01
.:9810 85 28 8D 04 D4 A9 C0 85
.:9818 69 60 A9 29 8D 05 D4 A9
.:9820 A5 8D 06 D4 A9 02 8D 01
.:9828 D4 A9 30 8D 00 D4 A5 28
.:9830 29 07 09 20 85 28 A5 28
.:9838 29 FE 8D 04 D4 09 01 85
.:9840 28 8D 04 D4 A9 C0 85 69
.:9848 60 A9 29 8D 05 D4 A9 85
.:9850 8D 06 D4 A9 15 8D 01 D4
.:9858 A9 30 8D 00 D4 A5 28 29
.:9860 07 09 80 85 28 A5 28 29
.:9868 FE 8D 04 D4 09 01 85 28
.:9870 8D 04 D4 A9 80 85 6A 60
.:9878 A5 28 29 FE 85 28 60 A5
.:9880 29 29 FE 85 29 60 A5 2A
.:9888 29 FE 85 2A 60 78 A9 00
.:9890 85 8D A5 64 C5 61 F0 04
.:9898 B0 19 90 14 A5 63 C5 60
.:98A0 F0 04 B0 0F 90 0A A5 62
.:98A8 C5 5F F0 04 B0 05 90 00
.:98B0 4C 48 99 4C FE 99 78 A5
.:98B8 9F 29 01 F0 22 A2 00 A0
.:98C0 00 B9 4F 00 F0 19 B9 4F
.:98C8 00 38 E9 01 99 4F 00 9D
.:98D0 01 D0 B9 3C 00 9D 00 D0
.:98D8 E8 E8 C8 C0 06 D0 E2 A5
.:98E0 44 8D 10 D0 A2 00 A5 9F
.:98E8 29 03 D0 28 AE 21 D0 CA
.:98F0 8E 21 D0 A5 8D D0 1D A9
.:98F8 AC 8D 05 D4 A9 04 8D 06
.:9900 D4 A9 81 8D 04 D4 A9 05
.:9908 8D 01 D4 A9 68 8D 00 D4
.:9910 A9 01 85 8D EA A9 FF 8D
.:9918 1D D0 8D 17 D0 A9 FB 8D
.:9920 12 D0 AD 11 D0 29 7F 8D
.:9928 11 D0 A9 01 85 A0 E6 9F
.:9930 E6 9E A5 9E 49 3C D0 04
.:9938 85 9E E6 9D EA A9 FF 8D
.:9940 19 D0 68 A8 68 AA 68 40
.:9948 78 A9 00 85 44 8D F8 5F
.:9950 8D F9 5F A9 12 8D FA 5F
.:9958 A9 10 8D FB 5F A9 11 8D
.:9960 FC 5F A9 1F 8D 15 D0 A9
.:9968 03 8D 1C D0 A9 D0 85 4F
.:9970 85 50 85 51 85 52 85 53
.:9978 85 54 A9 30 85 3C A9 F0
.:9980 85 3D A9 63 85 3E A9 9D
.:9988 85 3F A9 CD 85 40 A9 00
```

```
.:9990 8D 04 D4 A9 01 8D 29 D0
.:9998 8D 2A D0 8D 2B D0 8D 0B
.:99A0 D4 A9 B6 8D 14 03 A9 98
.:99A8 8D 15 03 A9 FB 8D 12 D0
.:99B0 AD 11 D0 29 7F 8D 11 D0
.:99B8 58 A9 00 85 9D A9 00 85
.:99C0 A0 A5 9D C9 18 D0 08 A9
.:99C8 80 8D 04 D4 4C 7C 83 D0
.:99D0 0D A5 9F 29 03 D0 07 AE
.:99D8 21 D0 E8 8E 21 D0 A5 A0
.:99E0 AD 00 DC 2D 01 DC 29 10
.:99E8 85 9C A5 9D C9 04 90 07
.:99F0 A5 9C D0 03 4C 7C 83 A5
.:99F8 A0 F0 E3 4C BD 99 78 A9
.:9A00 00 85 AB A9 1C 85 AC A9
.:9A08 FF 8D 1D D0 A9 00 85 9D
.:9A10 8D 04 D4 8D 0B D4 8D 15
.:9A18 D0 85 9B A9 02 8D 01 D4
.:9A20 8D 00 D4 A9 CF 8D 05 D4
.:9A28 A9 00 8D 06 D4 A9 81 8D
.:9A30 04 D4 A9 4A 8D 14 03 A9
.:9A38 9B 8D 15 03 A9 FB 8D 12
.:9A40 D0 AD 11 D0 29 7F 8D 11
.:9A48 D0 58 A9 00 85 A0 A2 04
.:9A50 A0 00 B1 AB 10 43 C9 FF
.:9A58 D0 0F A9 01 85 9B A9 00
.:9A60 85 AB A9 1C 85 AC 4C 21
.:9A68 9B C8 B1 AB 85 8D C8 A9
.:9A70 D8 18 71 AB 85 8E A5 9B
.:9A78 D0 05 A9 04 4C 82 9A AD
.:9A80 21 D0 A0 00 91 8D A5 AB
.:9A88 18 69 03 85 AB A9 00 65
.:9A90 AC 85 AC CA D0 BA 4C 1A
.:9A98 9B 29 40 F0 3F C8 B1 AB
.:9AA0 85 8D C8 A9 5C 18 71 AB
.:9AA8 85 8E A0 00 B1 8D 29 F0
.:9AB0 85 95 A5 9B D0 07 A5 95
.:9AB8 09 04 4C C4 9A AD 21 D0
.:9AC0 29 0F 05 95 91 8D A5 AB
.:9AC8 18 69 03 85 AB A9 00 65
.:9AD0 AC 85 AC CA F0 03 4C 50
.:9AD8 9A 4C 1A 9B C8 B1 AB 85
.:9AE0 8D C8 A9 5C 18 71 AB 85
.:9AE8 8E A0 00 B1 8D 29 0F 85
.:9AF0 95 A5 9B D0 07 A5 95 09
.:9AF8 40 4C 05 9B AD 21 D0 0A
.:9B00 0A 0A 0A 05 95 91 8D A5
.:9B08 AB 18 69 03 85 AB A9 00
.:9B10 65 AC 85 AC CA F0 03 4C
.:9B18 50 9A A5 9B F0 03 4C 4E
.:9B20 9A A5 9D C9 1A D0 03 4C
.:9B28 7C 83 A5 A0 AD 00 DC 2D
.:9B30 01 DC 29 10 85 9C A5 9D
.:9B38 C9 04 90 07 A5 9C D0 03
.:9B40 4C 7C 83 A5 A0 F0 E3 4C
.:9B48 4A 9A A5 9F 29 03 D0 0B
.:9B50 A5 9B F0 07 AC 21 D0 C8
.:9B58 8C 21 D0 A9 01 85 A0 E6
.:9B60 9E E6 9F A5 9E 49 3C D0
.:9B68 04 85 9E E6 9D EA A9 FB
.:9B70 8D 12 D0 AD 11 D0 29 7F
.:9B78 8D 11 D0 A9 FF 8D 19 D0
.:9B80 68 A8 68 AA 68 40 EA EA
```

Listing C-24: The BOGHOP Program, Source Code for the BOGHOP.O Program

```
1000 ;PUT"@0:BOGHOP"
1010 ;LOAD"ASM",8
1020 ;
1030 ;BOGHOP GAME
1040 ;
1050 ;TO START THE GAME FROM BASIC
1060 ;TYPE 'SYS 5120'
1070 ;
1080 ;CREATED 7/17/84
1090 ;(C) COPYRIGHT 1984, STEVEN BRESS
1100 ;
1110 ;LATEST ADDITIONS 11/15/84
1120 ;03:30 PM
1130 ;
```

```
1140          .OPT LIST,NOSYM,NOGEN
1150 ;
1160          .PAGE 'MACRO LIBRARY'
1170          .LIB MACLIB
1180          .PAGE 'SYSTEM DEFINITIONS'
1190          .LIB SYSDEF
1200          .PAGE 'RAM DEFINITIONS'
1210          .LIB BOGDEF
1220          .PAGE 'NOTE DEFINITIONS'
1230          .LIB COMMON
1240 ;
1250 *        = $1400
1260          JMP TOP
1270 ;
1280 ;LOOKUPS IN FILE LKUP
1290 ;
1300          .PAGE 'DATA SEGMENT'
1310          .LIB BOGDAT      ;DATA FILE
1320          .PAGE 'POINT PLOTTING'
1330          .LIB XXPLOT
1340          .PAGE 'BOG HOP CODE'
1350 ;
1360 ZERO    = $1042
1370 ONE     = $1052
1380 TWO     = $1062
1390 THREE   = $1072
1400 FOUR    = $1082
1410 FIVE    = $1092
1420 SIX     = $10A2
1430 SEVEN   = $10B2
1440 EIGHT   = $10C2
1450 NINE    = $10D2
1460 NUL     = $10E2
1470 VCLKLO  = $10F2
1480 VCLKHI  = $110B
1490 VLKUPL  = $1125
1500 VLKUPH  = $11FD
1510 HLKUPL  = $12CD
1520 HLKUPH  = $12F5
1530 ;
1540 ;
1550 RET      RTI
1560 TOP      SEI
1570 ;
1580          KILL
1590          ADRES INT0,CINV ;SET-UP 1'ST INTERRUPT
1600 ;
1610          LDMEM #BLACK,BORCL
1620          STA BCOL0
1630          BANK 1
1640          GRABAS $2000      ;SCREEN @ $6000
1650          TXBAS $1C00       ;TEXT @ $5C00
1660          ADRES $6000,GBASE
1670          GRAPH             ;BIT-MAPPED GRAPHICS ON
1680          MULTON            ;MULTI COLOR MODE ON
1690          MVCOL WHITE       ;CLEAR COLOR RAM TO WHITE
1700          FILL $6000,$00,$20 ;CLEAR GRAPHICS PAGE
1710          FILL $5C00,$56,$04 ;GREEN AND PURPLE
1720 ;
```

```
1730 ;
1740 ;
1750         LDMEM #$03,LIVES ;START WITH 3 LIVES
1760         STA COLOR
1770         LDMEM #$01,HORNC
1780         LDMEM #$01,VERNC
1790 ;
1800         LDMEM #$14,PLAYP ;PLAYER SPRITE POINTER
1810 ;
1820 ;
1830         LDMEM #$2A,MOUNTV ;MOUNTAIN VERTICAL SETTING
1840         STA MOUNTV+1
1850         STA MOUNTV+2
1860         STA MOUNTV+3
1870         LDMEM #$2D,MOUNTV+4
1880         STA MOUNTV+5
1890         STA MOUNTV+6
1900         STA MOUNTV+7
1910 ;
1920         LDMEM #$00,MOUNTH ;MOUNTAIN HORIZONTAL INITIAL POSITION
1930         LDMEM #$30,MOUNTH+1
1940         LDMEM #$60,MOUNTH+2
1950         LDMEM #$90,MOUNTH+3
1960         LDMEM #$C0,MOUNTH+4
1970         LDMEM #$F0,MOUNTH+5
1980         LDMEM #$20,MOUNTH+6
1990         LDMEM #$50,MOUNTH+7
2000 ;
2010         LDMEM #$09,MOUNTC ;MOUNTAIN COLOR
2020         STA MOUNTC+1
2030         STA MOUNTC+2
2040         STA MOUNTC+3
2050         STA MOUNTC+4
2060         STA"MOUNTC+5
2070         SVC MOUNTC+6
2080         STA MOUNTC+7
2090 ;
2100         LDMEM #$20,MOUNTP ;POINTER
2110         LDMEM #$21,MOUNTP+1
2120         LDMEM #$22,MOUNTP+2
2130         LDMEM03$23,MOUNTP+3
2140         LDMEM #$20,MOUNTP+4
2150         LDMEM #$21,MOUNTP+5
2160         LDMEM #$22,MOUNTP+6
2170         LDMEM #$23,MOUNTP+7
2180 ;
2190         LDMEM #WHITE,PLAYC ;PLAYER COLOR
2200         LDMEM #GREEN,MEANC
2210         STA MEANC+1
2220         STA MEANC+2
2230         STA MEANC+3
2240         STA MEANC+4
2250         STA MEANC+5
2260         STA MEANC+6
2270 ;
2280         LDMEM #RED,SPMCL0
2290         LDMEM #YELOW,SPMCL1
2300         LDMEM #$FF,BPRIOR
2310 ;
```

```
2320 ;
2330         LDMEM #$C0,MNTMSB ;SET THE MSB IN MOUNTH+7
2340 ;
2350 ;THE MOUNTAINS WILL BE SET UP
2360 ;DURING INT0
2370 ;
2380 ;INT1 WILL SET UP THE PLAIERS
2390 ;
2400 ;
2410         LDMEM #$00,SCORE ;CLEAR SCORE
2420         STA SCORE+1
2430         STA SCORE+2
2440         STA SCORE+3
2450 ;
2460         LDMEM #$01,LEVEL ;START @ LEVEL1
2470 ;
2480 ;SET SOUND GENERATOR 3 TO NOISE
2490 ;TO BE USED FOR RANDOM NUMBERS
2500 ;
2510         ADRES AN3,S2FRLO
2520         ADRES CN4,S1FRLO
2530         LDMEM #$0F,MMOD ;VOLUME=MAX
2540         LDMEM #$20,S1CORG ;TRIANGLE WAVE
2550         LDMEM #$28,S1ATDC
2560         LDMEM #$88,S1SURL
2570         LDA #<GN6
2580         STA S3FRLO
2590         LDA #>GN6
2600         STA S3FRHI
2610         LDMEM #$88,V3CORG
2620         LDMEM #$80,V3CORG ;RESET CHANNEL
2630         STA S3CORG
2640 ;
2650 ;
2660 ;INITIALIZE PLAYER POSITIONS
2670 ;
2680         LDMEM #$FF,WHOLIV
2690         LDMEM #$30,PLAYH
2700         LDMEM #$70,PLAYV
2710 ;
2720         LDMEM #$68,MEANV ;BAD GUY VERT
2730         LDMEM #$70,MEANV+1
2740         LDMEM #$80,MEANV+2
2750         LDMEM #$90,MEANV+3
2760         LDMEM #$A0,MEANV+4
2770         LDMEM #$B0,MEANV+5
2780         LDMEM #$C5,MEANV+6
2790 ;
2800         LDMEM #$90,MEANH ;BAD GUY HORIZ
2810         LDMEM #$95,MEANH+1
2820         LDMEM #$85,MEANH+2
2830         LDMEM #$90,MEANH+3
2840         LDMEM #$70,MEANH+4
2850         LDMEM #$75,MEANH+5
2860         LDMEM #$95,MEANH+6
2870 ;
2880 ;
2890         ADRES BEE1,MEANMV
2900         ADRES BEE2,MEANMV+2
```

```
2910         ADRES BEE3,MEANMV+4
2920         ADRES BEE4,MEANMV+6
2930         ADRES BEE1,MEANMV+8
2940         ADRES BEE2,MEANMV+$A
2950         ADRES BEE3,MEANMV+$C
2960 ;
2970 ;
2980         LDA SSCOL         ;CLEAR COLLISION REGISTERS
2990         LDA SBCOL
3000 ;
3010 ;
3020 ;
3030 ;
3040         RAST #$FB
3050         CLI
3060 ;
3070 START   ANOP              ;BEGINNING OF PROGRAM
3080 ;
3090         LDA LIVES         ;SEE IF GAME OVER
3100         ORA PLAYE
3110         ORA MEANE
3120         ORA MEANE+1
3130         ORA MEANE+2
3140         ORA MEANE+3
3150         ORA MEANE+4
3160         ORA MEANE+5
3170         ORA MEANE+6
3180         JEQ STWID         ;PASS OVER CODE IF OVER
3190 ;
3200 ;
3210 ;UPDATE THE SCORE
3220 ;SUPPRESS LEADING ZEROS
3230 ;
3240         LDA SCREEN        ;DON'T UPDATE SCORE
3250         AND #$0F          ;VERY OFTEN
3260         CMP #$0F          ;4 TIMES/SECOND
3270         JNE NOSCR1
3280 ;
3290 ;
3300         LDMEM #$0A,BUF    ;CLEAR TO NUL
3310         STA BUF+1
3320         STA BUF+2
3330         STA BUF+3
3340         STA BUF+4
3350         STA BUF+5
3360         STA BUF+6
3370         STA BUF+7
3380 ;
3390         LDA SCORE+3
3400         AND #$F0          ;MASK HIGH BYTE
3410         NIBLR             ;SHIFT DATA RIGHT
3420         STA BUF           ;PUT IN BUFFER
3430 ;
3440 NXDIG1  LDA SCORE+3
3450         AND #$0F          ;MASK LOW BYTE
3460         STA BUF+1         ;NEXT BUFFER POSITION
3470 ;
3480 NXDIG2  LDA SCORE+2
3490         AND #$F0
```

```
3500        NIBLR
3510        STA BUF+2
3520 ;
3530 NXDIG3 LDA SCORE+2
3540        AND #$0F
3550        STA BUF+3
3560 ;
3570 NXDIG4 LDA SCORE+1
3580        AND #$F0
3590        NIBLR
3600        STA BUF+4
3610 ;
3620 NXDIG5 LDA SCORE+1
3630        AND #$0F
3640        STA BUF+5
3650 ;
3660 NXDIG6 LDA SCORE
3670        AND #$F0
3680        NIBLR
3690        STA BUF+6
3700 ;
3710 NXDIG7 LDA SCORE
3720        AND #$0F
3730        STA BUF+7        ;DIGIT
3740 ;
3750 ;
3760 ;LEADING ZERO SUPPRESSION
3770 ;
3780        LDA BUF+0
3790        BNE NZSUP        ;IF NOT 0--NO SUPPRESS
3800 ;
3810        LDMEM #$0A,BUF+0 ;PUT IN NUL
3820 ;
3830        LDA BUF+1
3840        BNE NZSUP
3850 ;
3860        LDMEM #$0A,BUF+1
3870 ;
3880        LDA BUF+2
3890        BNE NZSUP
3900 ;
3910        LDMEM #$0A,BUF+2
3920 ;
3930        LDA BUF+3
3940        BNE NZSUP
3950 ;
3960        LDMEM #$0A,BUF+3
3970 ;
3980        LDA BUF+4
3990        BNE NZSUP
4000 ;
4010        LDMEM #$0A,BUF+4
4020 ;
4030        LDA BUF+5
4040        BNE NZSUP
4050 ;
4060        LDMEM #$0A,BUF+5
4070 ;
4080        LDA BUF+6
```

```
4090         BNE NZSUP
4100 ;
4110         LDMEM #$0A,BUF+6
4120 ;
4130 NZSUP   ANOP               ;ALL LEADING 0=NUL
4140 ;
4150 ;
4160 ;
4170 ;
4180 ;THE SCORES ARE IN BUF TO BUF+7
4190 ;
4200         LDX #$00
4210         LDMEM #$00,LPCNT1
4220         ADRES $6000,BUF1+2
4230 MKSCR1  LDA BUF,X
4240         ASL A
4250         TAY
4260         LDA NUMBER,Y
4270         STA BUF1
4280         LDA NUMBER+1,Y
4290         STA BUF1+1
4300         LDY #$00
4310 MKSCR2  LDA (BUF1),Y
4320         STA (BUF1+2),Y
4330         INY
4340         CPY #$10
4350         BNE MKSCR2
4360 ;
4370         DBINC BUF1+2,$10
4380 ;
4390         INX
4400         CPX #$08
4410         BNE MKSCR1         ;REPEAT FOR 8 DIGITS
4420 ;
4430 ;
4440 NOSCR1  ANOP
4450 ;
4460 ;SCORES ARE ON SCREEN
4470 ;
4480 ;
4490 ;
4500 ;MOVE THE CURRENT BAD GUYS
4510 ;
4520 ;
4530         LDA SCREEN         ;MOVE THE BAD GUYS ON
4540         AND #$01           ;EVERY OTHER SCREEN
4550         JEQ MVEM8
4560 ;
4570 ;
4580         LDX #$00
4590         LDY #$00
4600         STY LPCNT1
4610 ;
4620 MVEM    ANOP
4630         LDA WHOLIV
4640         LDY LPCNT1
4650         AND BITPOS,Y
4660         BEQ MVEM3
4670         LDA MEANE,Y
```

```
4680         BNE MVEM3
4690 ;
4700         LDA (MEANMV,X)   ;GET NEXT MOVEMENT BYTE
4710         BNE MVEM1        ;GOOD CODE--MOVE
4720 ;
4730 ;CHOOSE A NEW DIRECTION
4740 ;
4750         LDY LEVEL
4760         LDA BADSEQ,Y
4770         ASL A
4780         TAY
4790         LDA BASLK,Y      ;THE BASE ADDRESS OF
4800         STA BUF          ;THE LOOKUP TABLE OF
4810         LDA BASLK+1,Y    ;MOVEMENT PATTERNS
4820         STA BUF+1        ;IS IN BUF
4830 ;
4840         LDA RAND1        ;GET A RANDOM NUMBER
4850         AND #$06         ;TO CHOOSE A NEW MOVEMENT PATTERN
4860         TAY
4870 ;
4880         LDA (BUF),Y
4890         STA MEANMV,X     ;PUT THE NEW ADDRESS INTO
4900         INY              ;THE MOVEMENT REGISTERS
4910         LDA (BUF),Y
4920         STA MEANMV+1,X
4930         TXA
4940         ADC RAND1
4950         EOR SCREEN
4960         AND #$07
4970         ADC MEANMV,X
4980         STA MEANMV,X
4990         LDA #$00
5000         ADC MEANMV+1,X
5010         STA MEANMV+1,X
5020 ;
5030         LDA (MEANMV,X)   ;GET THE DIRECTION BYTE
5040 ;
5050 MVEM1   STA BUF1         ;HIDE THE DIRECTION
5060         LDY LEVEL
5070         LDA SPEEDH,Y
5080         STA HORNC        ;GET THE HORIZONTAL
5090         LDA SPEEDV,Y     ;AND VERTICAL SPEED
5100         STA VERNC
5110 ;
5120         LDY LPCNT1
5130 ;
5140         LDA MEAND,Y
5150         BEQ MVEM2        ;IF DIRECTION=1 THEN
5160 ;
5170         LDA BUF1         ;INVERT THE BITS
5180         NOT
5190         STA BUF1
5200 ;
5210 MVEM2   LDA BUF1         ;COPY THE BYTE TO
5220         STA BUF1+1       ;BUF1+1
5230 ;
5240         ROR BUF1
5250         BCC MVEM4
5260         JSR UP
```

```
5270 ;
5280 MVEM4   ROR BUF1
5290         BCC MVEM5
5300         JSR DN
5310 ;
5320 MVEM5   ROR BUF1
5330         BCC MVEM6
5340         JSR LT
5350 ;
5360 MVEM6   ROR BUF1
5370         BCC MVEM3
5380         JSR RT
5390 ;
5400 MVEM3   ANOP
5410         LDA MEANMV,X      ;INCREMENT MOVEMENT
5420         CLC
5430         ADC #$01
5440         STA MEANMV,X
5450         LDA MEANMV+1,X
5460         ADC #$00
5470         STA MEANMV+1,X
5480 ;
5490         INX
5500         INX
5510         INC LPCNT1        ;BUMP LOOP COUNTER
5520         LDA LPCNT1
5530         CMP #$07          ;DO IT 7 TIMES
5540         JNE MVEM
5550 ;
5560 MVEM8   ANOP
5570 ;
5580 ;DONE MOVING HERE
5590 ;
5600 ;
5610 ;CHECK AND INCREMENT LEVEL NEXT
5620 ;WHEN WHOLIV=0--GOTO NEXT LEVEL
5630 ;
5640         LDA WHOLIV
5650         AND #$7F          ;PLAYER DOESN'T COUNT
5660         JNE LEVDN         ;BRANCH TO LEVEL CHECK DONE
5670 ;
5680         LDMEM #$FF,WHOLIV
5690 ;
5700 ;RESTART BAD GUYS POSITIONS
5710 ;
5720         LDX #$00
5730         LDMEM #$70,BUF    ;VERTICAL OFFSET
5740         LDA RAND1         ;GET A RANDOM NUMBER
5750         AND #$7F          ;NUMBER TO ADD TO OFFSET
5760         ADC BUF
5770         STA MEANV         ;FIRST VERTICAL
5780 ;
5790         LDA RAND2
5800         AND #$7F
5810         ADC BUF
5820         STA MEANV+1
5830 ;
5840         LDA RAND3
5850         AND #$7F
```

```
5860         ADC BUF
5870         STA MEANV+2
5880 ;
5890         LDA RAND4
5900         ADC #$7F
5910         ADC BUF
5920         STA MEANV+3
5930 ;
5940         LDA RAND1
5950         EOR RAND2          ;MODIFY RANDOM #
5960         AND #$7F
5970         ADC BUF
5980         STA MEANV+4
5990 ;
6000         LDA RAND2
6010         EOR RAND3
6020         AND #$7F
6030         ADC BUF
6040         STA MEANV+5
6050 ;
6060         LDA RAND3
6070         EOR RAND1
6080         AND #$7F
6090         ADC BUF
6100         STA MEANV+6
6110 ;
6120         LDMEM #$90,MEANH ;HORIZONTALS
6130         LDMEM #$85,MEANH+1
6140         LDMEM #$95,MEANH+2
6150         LDMEM #$70,MEANH+3
6160         LDMEM #$75,MEANH+4
6170         LDMEM #$90,MEANH+5
6180         LDMEM #$80,MEANH+6
6190 ;
6200         LDMEM #$00,MEAND ;CLEAR DIRECTIONS
6210         STA MEAND+1
6220         STA MEAND+2
6230         STA MEAND+3
6240         STA MEAND+4
6250         STA MEAND+5
6260         STA MEAND+6
6270 ;
6280         ADRES NUL,MEANMV ;SET MOVEMENT DIRECTION
6290         ADRES NUL,MEANMV+2 ;POINTERS TO POINT
6300         ADRES NUL,MEANMV+4 ;AT KNOWN ZEROS
6310         ADRES NUL,MEANMV+6 ;CAUSING NEW DIRECTIONS
6320         ADRES NUL,MEANMV+8 ;TO BE CHOSEN
6330         ADRES NUL,MEANMV+$0A
6340 ;
6350         LDX LEVEL          ;GET LEVEL
6360         LDA BADSEQ,X       ;FIND THE NEXT BAD GUY
6370         ASL A              ;MULTIPLY BY 4
6380         ASL A              ;USE AS A SPRITE POINTER BASE
6390         TAX
6400         STX MEANP          ;DON'T ALL FLAP AT ONCE
6410         STX MEANP+6
6420         INX
6430         STX MEANP+1
6440         STX MEANP+3
```

```
6450         INX
6460         STX MEANP+2
6470         INX
6480         STX MEANP+4
6490         STX MEANP+5
6500 ;
6510 ;
6520 ;ALL DONE WITH RESETTING
6530 ;
6540         INC LEVEL
6550         LDA LEVEL
6560         CMP #$12         ;MAXIMUM DEFINED LEVEL
6570         BCC LEVDN
6580 ;
6590         DEC LEVEL        ;LEVEL TOO HIGH
6600 ;
6610 LEVDN   ANOP
6620 ;
6630 ;CHECK THE SCREEN POSITION
6640 ;TO CHANGE DIRECTION
6650         LDX #$00
6660         LDY #$00         ;CLEAR INDEXES
6670 ;
6680 DIRCK1  LDA MEANH,X
6690         CMP #$16         ;REVERSE DIRECTION
6700         BCS DIRCK2       ;IF AT THE LEFT SIDE
6710 ;
6720         LDA #$01         ;OF THE SCREEN
6730         STA MEAND,X
6740 ;
6750 DIRCK2  CMP #$98         ;CHECK RIGHT SIDE
6760         BCC DIRCK3
6770 ;
6780         LDA #$00
6790         STA MEAND,X
6800 ;
6810 DIRCK3  INX
6820         CPX #$07         ;REPEAT FOR ALL 7
6830         BNE DIRCK1
6840 ;
6850 ;
6860 ;
6870 ;CHECK COLLISIONS BETWEEN THE
6880 ;PLAYER'S SHOT AND THE BAD GUYS
6890 ;NO CHECK FOR A DEAD BAD GUY
6900 ;
6910 ;THE PLAYER AND HIS SHOT ARE
6920 ;MULTIPLEXED WITH THE SHOT ON
6930 ;ODD SCREENS
6940 ;
6950         LDA SCREEN
6960         AND #$01         ;THE SHOT ISN'T UP IF 1
6970         JNE BCOLND       ;NO BAD GUYS HIT--
6980 ;
6990 ;FIRST FIND IF THERE IS A HIT
7000 ;THEN FIND WHO WAS HIT
7010 ;
7020         LDA TMSCOL
7030         AND #$01         ;WAS THE SHOT HIT
```

```
7040        JEQ BCOLND       ;BY A BAD GUY
7050 ;
7060 ;A HIT IF HERE--SEE IF ALIVE
7070 ;
7080        LDA WHOLIV
7090        ASL A            ;POSITION BITS TO MATCH COLLISION REGISTER
7100        STA BUF
7110        LDA TMSCOL
7120        AND #$FE         ;CLEAR PLAYER SHOT
7130        AND BUF          ;IF ZERO---NO HIT
7140        BEQ BCOLND
7150 ;
7160 ;FIND THE HIT HERE
7170 ;
7180        LSR A
7190        STA BUF
7200 ;
7210        LDX #$00
7220 BHIT0  ANOP
7230        ROR BUF
7240        BCC NOBCOL       ;IF CLEAR--NOT HIT
7250        LDA MEANE,X      ;SEE IF EXPLODING
7260        BNE NOBCOL       ;NEXT IF IT IS
7270 ;
7280 ;
7290 ;IF HERE--MIGHT BE HIT
7300 ;
7310        LDA MEANV,X      ;SEE IF THE VERT IS IN RANGE
7320        CLC
7330        ADC #$10         ;ADD AN OFFSET
7340        SEC
7350        SBC SHOTSV+0     ;SUBTRACT PLAYER SHOT
7360        CMP #$10         ;IF THE RESULT IS NOT
7370        BCS NOBCOL       ;IN RANGE--BRANCH
7380 ;
7390 ;VERTICAL IS IN RANGE, CHECK HORIZONTAL
7400 ;
7410        LDA MEANH,X
7420        CLC
7430        ADC #$0C
7440        SEC
7450        SBC SHOTSH+0
7460        CMP #$0C
7470        BCS NOBCOL
7480 ;
7490 ;IT'S A HIT IF HERE
7500 ;
7510 ;START EXPLOSION
7520 ;
7530 ;
7540 ;START EXPLOSION TIMER
7550 ;
7560        LDA #$C0         ;ABOUT 3 SECONDS
7570        STA MEANE,X
7580        LDA #$18
7590        STA MEANP,X
7600 ;
7610 ;
7620 ;EXPLODE HERE WITH SOUND
```

```
7630 ;
7640 ;
7650        ADRES GN6,S2FRLO ;HI EXPLOSION TONE
7660        LDMEM #$05,S2ATDC
7670        LDMEM #$00,S2SURL
7680        LDMEM #$80,V2CORG ;CLEAR SOUND CHANNEL
7690        LDMEM #$81,S2CORG ;ENABLE CHANNEL
7700        LDMEM #$18,SNDTM2
7710 ;
7720 ;ADD TO SCORE HERE
7730 ;
7740 ;100 POINTS + 8*LEVEL
7750 ;
7760        LDA LEVEL
7770        ASL A            ;MULTIPLY LEVEL BY 8
7780        ASL A
7790        ASL A
7800 ;
7810        SED              ;SET DECIMAL ARITHMETIC MODE
7820        CLC              ;CLEAR THE CARRY BIT
7830        ADC SCORE+0
7840        STA SCORE+0
7850        LDA SCORE+1
7860        ADC #$01         ;ADD THE 100 POINTS
7870        STA SCORE+1
7880        LDA SCORE+2
7890        ADC #$00
7900        STA SCORE+2
7910        LDA SCORE+3
7920        ADC #$00
7930        STA SCORE+3
7940        CLD
7950 ;
7960 ;KILL SHOT AFTER A HIT
7970 ;
7980        LDA SHOTS
7990        AND #$FE
8000        STA SHOTS
8010        JMP BCOLND
8020 ;
8030 ;
8040 ;
8050 ;
8060 NOBCOL INX
8070        CPX #$07
8080        BNE BHIT0
8090 ;
8100 BCOLND ANOP
8110 ;
8120 ;
8130 ;SEE IF PLAYER WAS HIT
8140 ;THE PLAYER IS ON EVEN SCREENS
8150 ;
8160        LDA SCREEN
8170        AND #$01
8180        BEQ NOPCOL
8190 ;
8200        LDA TMBCOL
8210        AND #$01
```

```
8220        BEQ NOPCOL
8230 ;
8240 ;PLAYER HIT IF HERE
8250 ;PRETTY EASY WHEN THE HARDWARE
8260 ;CAN CHECK COLLISIONS FOR YOU!!!
8270 ;
8280        LDA LIVES       ;IF DEAD, DON'T KILL
8290        BEQ NOPCOL
8300 ;
8310        LDA PLAYE       ;MAKE SURE NOT ALREADY
8320        BNE NOPCOL      ;IN EXPLOSION
8330 ;
8340        DEC LIVES
8350 ;
8360        LDMEM #$C0,PLAYE ;START EXPLODE
8370        LDMEM #$18,PLAYP ;POINT AT EXPLOSION
8380        LDMEM #$30,SNDTM1
8390        LDMEM #$80,V1CORG ;RESET CHANNEL
8400        LDMEM #$1A,S1ATDC
8410        LDMEM #$81,S1CORG
8420        ADRES EN5,S1FRLO ;HIGH EXPLOSION
8430 ;
8440 ;
8450 ;
8460 NOPCOL ANOP
8470 ;
8480 ;
8490 ;
8500 ;LAUNCH SHOTS NEXT
8510 ;
8520        LDA SCREEN
8530        AND #$03
8540        BNE NOLNCH
8550 ;
8560 ;TRY TO LAUNCH A SHOT
8570 ;
8580        LDX #$00
8590        LDA SHOTS       ;ONE SHOT IN THE AIR
8600        NOT             ;AT ANY ONE TIME
8610        LSR A
8620        AND WHOLIV      ;ONLY LIVE BAD GUYS
8630        STA BUF         ;CAN SHOOT
8640 ;
8650 LCHIT1 ROR BUF
8660        BCC LCHIT2
8670 ;
8680        LDY LEVEL
8690        LDA FIRPOW,Y
8700        CMP RAND2       ;IF RANDOM #<FIRPOW--SHOOT
8710        BCC NOLNCH
8720 ;
8730        LDA MEANE,X
8740        BNE LCHIT2
8750 ;
8760 ;
8770 ;FIRE IF HERE
8780 ;
8790        LDA SHOTS
8800        ORA BITPOS+1,X
```

```
8810         STA SHOTS
8820 ;
8830         LDA MEANV,X
8840         CLC
8850         ADC #$08         ;ADD AN OFFSET
8860         STA SHOTSV+1,X
8870 ;
8880         LDA MEANH,X
8890         STA SHOTSH+1,X
8900 ;
8910         LDA RAND3        ;PICK A RANDOM DIRECTION
8920         AND #$03         ;FOR THE SHOT
8930         TAY
8940         LDA SHTDR,Y
8950         STA SHOTSD+1,X
8960         STX BUF1         ;PUT X IN TEMP REGISTER
8970         LDA SHOTSH+1,X
8980         STA XPNT
8990         LDA SHOTSV+1,X
9000         STA YPNT
9010         JSR XPLOT        ;PLOT THE SHOT
9020 ;
9030 ;
9040 ;
9050 ;PUT SOUND HERE
9060 ;
9070 ;
9080         ADRES EN4,S2FRLO
9090         TXA              ;GET BAD GUY NUMBER
9100         CLC              ;PREPARE TO ADD
9110         ADC S2FRHI       ;HIGHER TONE FOR
9120         STA S2FRHI       ;HIGHER NUMBER
9130 ;
9140         LDMEM #$20,V2CORG ;SAWTOOTH WAVE
9150         LDMEM #$09,S2ATDC
9160         LDMEM #$00,S2SURL
9170         LDMEM #$21,S2CORG
9180         LDMEM #$14,SNDTM2
9190         JMP NOLNCH
9200 ;
9210 ;ONCE A SHOT HAS BEEN ENABLED
9220 ;IT IS MOVED AUTOMATICALLY
9230 ;BY THE MOVE SHOT ROUTINE
9240 ;
9250 LCHIT2 INX
9260         CPX #$07
9270         BNE LCHIT1
9280 ;
9290 NOLNCH ANOP
9300 ;
9310 ;
9320 ;
9330 ;
9340 ;UNPLOT THE SHOTS, THEN MOVE
9350 ;
9360 ;
9370         LDA SCREEN
9380         AND #$01
9390         JEQ NSHT0
```

```
9400         LDMEM #$03,COLOR
9410         LDX #$01
9420         STX LPCNT1
9430 ;
9440         LDMEM SHOTS,BUF
9450         ROR BUF
9460 ;
9470 MVSHT0  ANOP
9480         LDX LPCNT1
9490         ROR BUF          ;SEE IF SHOT IN FLIGHT
9500         BCC MVSHT4
9510 ;
9520         LDA SHOTSH,X
9530         STA XPNT
9540         LDA SHOTSV,X
9550         STA YPNT
9560 ;
9570         JSR XPLOT        ;UNPLOT THE POINT
9580 ;
9590         LDX LPCNT1
9600 ;
9610         LDY LEVEL
9620 ;
9630         LDA SHOTSD,X
9640         STA BUF1
9650         ROR BUF1
9660         BCC MVSHT1
9670 ;
9680         LDA SHOTSV,X     ;MOVE UP
9690         CLC              ;ADD IN ONE TO SPEED
9700         SBC SPEEDV,Y
9710         STA SHOTSV,X
9720 ;
9730 MVSHT1  ROR BUF1
9740         BCC MVSHT2
9750 ;
9760         LDA SHOTSV,X
9770         SEC
9780         ADC SPEEDV,Y
9790         STA SHOTSV,X
9800 ;
9810 MVSHT2  ROR BUF1
9820         BCC MVSHT3
9830 ;
9840         LDA SHOTSH,X
9850         CLC
9860         SBC SPEEDH,Y
9870         STA SHOTSH,X
9880 ;
9890 MVSHT3  ANOP
9900 ;
9910 ;PREPARE TO PLOT
9920 ;
9930         LDA SHOTSH,X
9940         STA XPNT
9950         LDA SHOTSV,X
9960         STA YPNT
9970 ;
9980         JSR XPLOT
```

```
9990 ;
10000 MVSHT4 ANOP
10010 ;
10020          INC LPCNT1
10030          LDA LPCNT1
10040          CMP #$08
10050          JNE MVSHT0
10060 ;
10070 NSHT0   ANOP
10080 ;
10090 ;MOVE PLAYER SHOT
10100 ;
10110          LDA SHOTS
10120          AND #$01
10130          BEQ NSHT1        ;IF SHOT IS INFLIGHT
10140 ;
10150          INC SHOTSH       ;MOVE SHOT TO RIGHT
10160          INC SHOTSH
10170 ;
10180 NSHT1   ANOP
10190 ;
10200 ;
10210 ;CHECK TO SEE IF THE SHOTS HAVE
10220 ;REACHED THE EDGE--CANCEL SHOT
10230 ;WHEN EDGE IS REACHED
10240 ;
10250 ;CHECK PLAYER'S SHOT FIRST
10260 ;
10270          LDA SHOTS
10280          AND #$01         ;IS PLAYER SHOT ACTIVE?
10290          BEQ NSHT2
10300 ;
10310          LDA SHOTSH
10320          CMP #$A0         ;RIGHT EDGE OF SCREEN
10330          BCC NSHT2
10340 ;
10350          LDA SHOTS        ;CLEAR SHOT ENABLE BIT
10360          AND #$FE
10370          STA SHOTS
10380 ;
10390 NSHT2   ANOP
10400 ;
10410 ;CHECK THE BAD GUYS SHOTS
10420 ;
10430          LDX #$01
10440 NSHT3   LDA SHOTSH,X
10450          CMP #$0C         ;LEFT SIDE
10460          BCS NSHT4        ;NOT THERE YET
10470 ;
10480          JMP CLRSHT       ;UNPLOT AND CLEAR ENABLE
10490 ;
10500 NSHT4   LDA SHOTSV,X
10510          CMP #$F4
10520          BCC NSHT5
10530 ;
10540          JMP CLRSHT
10550 ;
10560 NSHT5   CMP #$50         ;ADJUST LATER----
10570          BCS NSHT6
```

```
10580 ;
10590 CLRSHT LDA SHOTS
10600        AND ANDPOS,X
10610        STA SHOTS
10620 ;
10630 ;UNPLOT THE SHOT
10640 ;
10650        STX BUF          ;SAVE X REGISTER
10660        LDA SHOTSH,X
10670        STA XPNT
10680        LDA SHOTSV,X
10690        STA YPNT
10700 ;
10710        JSR XPLOT        ;UNPLOT THE POINT
10720 ;
10730        LDX BUF          ;GET THE X BACK
10740 ;
10750 NSHT6  ANOP
10760        INX
10770        CPX #$08
10780        BNE NSHT3
10790 ;
10800 ;
10810 ;
10820 ;
10830 ;
10840 ;ANIMATE SPRITES NEXT
10850 ;ALL SPRITES USE A FOUR SPRITE
10860 ;ANIMATION SEQUENCE. 4/SECOND
10870 ;
10880 ;
10890        LDA SCREEN
10900        AND #$07         ;DO IT EVERY 15'TH
10910        CMP #$06         ;SCREEN
10920        BNE NOAN1        ;BYPASS CODE
10930 ;
10940        LDX #$00
10950 ;
10960 ANSP0  ANOP
10970        LDA PLAYP,X      ;SPRITE POINTER
10980        AND #$03         ;STRIP LOWER 3 BITS
10990        TAY
11000        LDA PLAYP,X
11010        AND #$FC         ;CLEAR 2 LSB'S
11020        STA BUF
11030        INY
11040        TYA
11050        AND #$03         ;CLEAR OVERFLOW
11060        ORA BUF
11070        STA PLAYP,X
11080 ;
11090        INX
11100        CPX #$08
11110        BNE ANSP0
11120 ;
11130 NOAN1  ANOP
11140 ;
11150 ;
11160        LDA SCREEN
```

```
11170          AND #$0F
11180          BNE NOLV
11190          ADRES $60C0,BUF
11200 ;
11210          LDA LIVES
11220          ASL A
11230          TAY
11240          LDA NUMBER,Y
11250          STA BUF+2
11260          LDA NUMBER+1,Y
11270          STA BUF+3
11280          LDY #$00
11290 LIVLP    LDA (BUF+2),Y
11300          STA (BUF),Y
11310          INY
11320          CPY #$10
11330          BNE LIVLP
11340 ;
11350 NOLV     ANOP
11360 ;
11370 ;
11380 ;
11390 ;
11400 ;
11410 ;
11420 ;
11430 STWID    LDA JOY2           ;RESET
11440          AND #$10
11450          JEQ TOP
11460 ;
11470 ;
11480 ;
11490 ;
11500 ;
11510 ;
11520 ;
11530 ;
11540          LDMEM #$00,ENABLE
11550 TWID     LDA ENABLE
11560          BEQ TWID
11570          JMP START
11580 ;
11590 ;
11600 ;
11610 INT0     ANOP
11620          LDA SSCOL          ;GET SPRITE COLLISIONS
11630          STA TMSCOL         ;PUT IN TEMP REGISTER
11640          LDA SBCOL          ;SPRITE--BACKGROUND
11650          STA TMBCOL         ;COLLISIONS
11660          LDMEM #BLUE,BCOL0
11670          INC SCREEN
11680          INC RANSEC
11690          LDA RANSEC
11700          EOR #$3C
11710          BNE SYNC1
11720 ;
11730          STA RANSEC
11740          INC SECOND
11750 SYNC1    ANOP
```

```
11760 ;
11770          LDX #$00
11780          STX HMSB
11790          LDY #$00
11800 SHAD1    ANOP
11810          LDA MOUNTV,X
11820          STA SPR0Y,Y
11830          LDA MOUNTP,X
11840          STA SPRPT1,X
11850          LDA MOUNTH,X
11860          STA SPR0X,Y
11870          LDA MOUNTC,X
11880          STA SPRCL0,X
11890 ;
11900 ;
11910 SHAD2    INY
11920          INY
11930          INX
11940          CPX #$08
11950          BNE SHAD1
11960 ;
11970          LDA MNTMSB
11980          STA XMSB
11990 ;
12000          LDMEM #$00,MLTSP
12010 ;
12020          LDMEM #$FF,SPRXSZ
12030          STA SPRYSZ
12040          STA SPREN
12050 ;
12060          LDA PLAYE
12070          BEQ TM0
12080          DEC PLAYE
12090          LDA PLAYE
12100          BNE TM0
12110          LDMEM #$14,PLAYP
12120 ;
12130 TM0      LDA SNDTM1
12140          BEQ TM1
12150          DEC SNDTM1
12160          BNE TM1
12170 ;
12180          LDA S1CORG
12190          AND #$FE
12200          STA S1CORG
12210 ;
12220 TM1      LDA SNDTM2
12230          BEQ TM2
12240          DEC SNDTM2
12250          BNE TM2
12260 ;
12270          LDA S2CORG
12280          AND #$FE
12290          STA S2CORG
12300 ;
12310 TM2      LDA SNDTM3
12320          BEQ TM3
12330          DEC SNDTM3
12340          BNE TM3
```

```
;
;
;
TM3     ANOP
;
;
;
;SHADOW SID CHIP
;
        LDMEM S1ATDC,V1ATDC
        LDMEM S2ATDC,V2ATDC
        LDMEM S3ATDC,V3ATDC
        LDMEM S1SURL,V1SURL
        LDMEM S2SURL,V2SURL
        LDMEM S3SURL,V3SURL
        LDMEM S1FRLO,V1FRLO
        LDMEM S2FRLO,V2FRLO
        LDMEM S3FRLO,V3FRLO
        LDMEM S1FRHI,V1FRHI
        LDMEM S2FRHI,V2FRHI
        LDMEM S3FRHI,V3FRHI
        LDMEM S1PWLO,V1PWLO
        LDMEM S2PWLO,V2PWLO
        LDMEM S3PWLO,V3PWLO
        LDMEM S1PWHI,V1PWHI
        LDMEM S2PWHI,V2PWHI
        LDMEM S3PWHI,V3PWHI
        LDMEM FILLO,FLCNLO
        LDMEM FILHI,FLCNHI
        LDMEM MMOD,MODVOL
        LDMEM RFIL,RESFLT
        LDMEM S1CORG,V1CORG
        LDMEM S2CORG,V2CORG
        LDMEM S3CORG,V3CORG
;
;
;GENERATE A RANDOM NUMBER
;USE RAND1-RAND4 BASED AN SCREEN
;
        LDA SCREEN
        AND #$03
        TAX             ;POINT TO A RAND REGISTER
        LDA RANDOM      ;THE FOLLOWING
        ADC RAND1       ;SEQUENCE THROWS
        ADC RAND3       ;A NUMBER OF RANDOM
        EOR RAND2       ;NUMBERS TOGETHER
        SBC RAND4       ;TO FORM A NEW RANDOM
        ADC RANSEC      ;NUMBER
        STA RAND1,X
;
;
;MOVE THE PLAYER USING JOY1
;
        LDA LIVES       ;SEE IF GAME OVER
        JEQ NOMOV4      ;DON'T MOVE IF OVER
;
        LDA PLAYE       ;DON'T MOVE IF EXPLODING
        JNE NOMOV4
;
```

```
12940          LDA JOY1
12950          NOT                  ;TURN THE BITS OVER
12960          STA IBUF+1           ;PUT THE DATA AWAY
12970          AND #$0F             ;CHECK TO SEE IF THE
12980          BEQ NOMOV3           ;BRANCH IF NOT
12990 ;
13000          LDA SCREEN           ;SLOW MOVEMENT BY
13010          AND #$01             ;MOVING ON ALTERNATE
13020          BNE NOMOV3           ;SCREENS
13030 ;
13040 ;IF HERE-MOVE CHARACTER
13050 ;
13060          LDMEM IBUF+1,IBUF ;COPY JOY TO BUF
13070          ROR IBUF             ;CHECK FIRST BIT
13080          BCC NXBIT1           ;IF CARRY CLEAR-JUMP
13090 ;
13100          LDA PLAYV
13110          SEC
13120          SBC #$01
13130          CMP #$68             ;CHECK FOR TOP
13140          BCC NXBIT1
13150 ;
13160          STA PLAYV
13170 ;
13180 NXBIT1   ROR IBUF
13190          BCC NXBIT2
13200 ;
13210          LDA PLAYV
13220          CLC
13230          ADC #$01
13240          CMP #$E8             ;CHECK FOR BOTTOM
13250          BCS NXBIT2
13260 ;
13270          STA PLAYV
13280 ;
13290 NXBIT2   ROR IBUF
13300          BCC NXBIT3           ;MOVE LEFT
13310 ;
13320          LDA PLAYH
13330          SEC
13340          SBC #$01
13350          CMP #$10
13360          BCC NXBIT3
13370 ;
13380          STA PLAYH
13390 ;
13400 NXBIT3   ROR IBUF
13410          BCC NOMOV3           ;MOVE RIGHT
13420 ;
13430          LDA PLAYH
13440          CLC
13450          ADC #$01
13460          CMP #$30
13470          BCS NOMOV3
13480 ;
13490          STA PLAYH
13500 ;
13510 NOMOV3   LDA JOY1
13520          AND #$10             ;CHECK FIRE BUTTON
```

```
        BNE NOMOV4       ;NOT PRESSED--BRANCH
;
        LDA SHOTS        ;SHOT INFLIGHT?
        AND #$01         ;PLAYER SHOT
        BNE NOMOV4       ;IF NOT--FIRE....
;
;
        LDA PLAYE
        BNE NOMOV4
;
        LDA SHOTS        ;SET SHOT IN FLIGHT BIT
        ORA #$01         ;FOR THE PLAYER
        STA SHOTS
;
;SET SHOT START POSITION TO GUN
;
        LDA PLAYH
        CLC
        ADC #$05         ;ADD OFFSET TO SHOT
        STA SHOTSH
;
        LDA PLAYV
        CLC
        ADC #$07         ;ADD VERTICLE OFFSET
        STA SHOTSV
;
        LDMEM #$08,SHOTSD ;SHOT DIRECTION
        LDMEM #$81,S1CORG
        LDMEM #$20,SNDTM1
        ADRES CN3,S1FRLO
        LDMEM #$28,S1ATDC
;
;SHOT STARTED
;
NOMOV4 ANOP
;
;
;
;
        ADRES INT1,CINV
        LDMEM #$01,ENABLE
        RAST #$54
        IPULL
;
;
;
INT1    SEI
;
        LDA SSCOL        ;CLEAR COLLISIONS
        LDA SBCOL
;
        NOP
        NOP
        NOP
;
;
        LDMEM #GREEN,BCOL0
;
        LDX #$00
```

```
        STX HMSB
        LDY #$00
SHAD3   ANOP
        LDA PLAYV,X
        STA SPR0Y,Y
        LDA PLAYP,X
        STA SPRPT1,X
        LDA PLAYH,X
        ASL A
        STA SPR0X,Y
        BCC SHAD4
        LDA HMSB
        ORA BITPOS,X
        STA HMSB
SHAD4   INY
        INY
        INX
        CPX #$08
        BNE SHAD3
;
;
        LDA HMSB
        STA XMSB
;
;MULTIPLEX PLAYER AND HIS SHOT
;THE PLAYER IS ALREADY UP
;
        LDA SCREEN      ;SHOT ON ODD SCREENS
        AND #$01
        BEQ NOMLT
;
        LDA SHOTS       ;SEE IF SHOT ACTIVE
        AND #$01        ;IF NOT ACTIVE-FORGET IT
        BEQ NOMLT
;
;SHOT ACTIVE IF HERE
;
        LDA #$1C
        STA SPRPT1
        LDA SHOTSV      ;GET SHOT VERTICAL
        STA SPR0Y       ;STORE IN SPRITE 0
        LDA SHOTSH      ;GET SHOT HORIZONTAL
        ASL A           ;MULTIPLY BY 2
        STA SPR0X       ;PUT IN VIC CHIP
        LDA XMSB        ;GET MSB
        AND #$FE        ;CLEAR MSB
        STA XMSB        ;REPLACE IT
        BCC NOMLT       ;GO HOME IF MSB CLEAR
;
        ORA #$01        ;SET MSB IF HERE
        STA XMSB
;
NOMLT   ANOP            ;MULTIPLEXING DONE
;
;
        LDMEM #$00,SPRXSZ ;SET SPRITE SIZE
        STA SPRYSZ      ;TO SMALL
;
        LDMEM #$FF,MLTSP ;MULTI COLOR SPRITES
```

```
14710 ;
14720         LDA WHOLIV       ;ENABLE THE LIVE BAD GUYS
14730         ASL A
14740         ORA #$01         ;ENABLE PLAYER ALWAYS
14750         STA SPREN        ;TURN SPRITES BACK ON
14760 ;
14770 ;
14780 ;THE MOUNTAINS ARE UP
14790 ;SO THEY CAN NOW BE MOVED
14800 ;
14810         LDX #$00
14820         LDMEM MNTMSB,HMSB
14830 MVMNT0 HDEC MOUNTH
14840         INX
14850         CPX #$08
14860         BNE MVMNT0
14870 ;
14880 LDMEM  HMSB,MNTMSB
14890 ;
14900 ;MOUNTAINS HAVE BEEN MOVED ONE
14910 ;PIXEL TO THE LEFT
14920 ;THEY MUST NOW BE ADJUSTED FOR
14930 ;A WRAPAROUND EFFECT TO APPEAR
14940 ;CONTINUOUS
14950 ;
14960         LDX #$00
14970 MVMNT1 ANOP
14980         PUNPCK MOUNTH,MNTMSB,IBUF
14990         LDA IBUF+1
15000         CMP #$01
15010         BNE MVMNT2
15020         LDA IBUF
15030         CMP #$E8
15040         BNE MVMNT2
15050 ;
15060 ;MOUNTAIN SECTION AT 0--MOVE IT
15070 ;TO $0170
15080 ;
15090         LDA MNTMSB
15100         ORA BITPOS,X
15110         STA MNTMSB
15120         LDA #$70
15130         STA MOUNTH,X
15140 ;
15150 MVMNT2 INX              ;DO THE NEXT SECTION
15160         CPX #$08
15170         BNE MVMNT1
15180 ;
15190 ;
15200 ;THE MOUNTAINS ARE FINISHED
15210 ;
15220         LDX #$00         ;DECREMENT EXPLOSION COUNTERS
15230 EXPLP1 LDA MEANE,X
15240         BEQ EXPLP2
15250         DEC MEANE,X
15260         BNE EXPLP2
15270 ;
15280 ;
15290 ;
```

```
15300 ;
15310        LDA WHOLIV
15320        AND ANDPOS,X     ;CLEAR BIT FROM WHOLIV
15330        STA WHOLIV
15340 ;
15350 EXPLP2 INX
15360        CPX #$07
15370        BNE EXPLP1
15380 ;
15390 ;
15400 ;
15410        ADRES INT0,CINV
15420        RAST #$FB
15430        IPULL
15440 ;
15450 ;
15460 ;
15470 ;
15480 UP     ANOP
15490        LDA MEANV,Y
15500        SEC
15510        SBC VERNC
15520        CMP #$68         ;UNDER MOUNTAINS
15530        BCC UP1
15540        STA MEANV,Y
15550 UP1    RTS
15560 ;
15570 DN     ANOP
15580        LDA MEANV,Y
15590        CLC
15600        ADC VERNC
15610        CMP #$E8
15620        BCS DWN1
15630        STA MEANV,Y
15640 DWN1   RTS
15650 ;
15660 LT     ANOP
15670        LDA MEANH,Y
15680        SEC
15690        SBC HORNC
15700        CMP #$0E
15710        BCC LT1
15720        STA MEANH,Y
15730 LT1    RTS
15740 ;
15750 RT     ANOP
15760        LDA MEANH,Y
15770        CLC
15780        ADC HORNC
15790        CMP #$A9
15800        BCS RT1
15810        STA MEANH,Y
15820 RT1    RTS
15830 ;
15840 ;
15850 ;
15860        .OPT LIST
15870        NOP
15880 ;
15890        .END
```

Listing C-25: The BOGHOP.O Program

```
B*
   PC  SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.

.:1000 28 43 29 20 43 4F 50 59
.:1008 52 49 47 48 54 20 31 39
.:1010 38 34 2C 20 53 54 45 56
.:1018 45 4E 20 42 52 45 53 53
.:1020 28 50 29 20 50 45 52 46
.:1028 4F 52 4D 41 4E 43 45 20
.:1030 31 39 38 34 2C 20 53 54
.:1038 45 56 45 4E 20 42 52 45
.:1040 53 53 3C C0 C3 CC F0 C0
.:1048 3F 00 00 C0 C0 C0 C0 C0
.:1050 00 00 0C 3C 0C 0C 0C 0C
.:1058 3F 00 00 00 00 00 00 00
.:1060 00 00 3F C0 00 0F 30 C0
.:1068 FF 00 00 C0 C0 00 00 00
.:1070 C0 00 FF 00 03 0F 00 C0
.:1078 3F 00 C0 C0 00 00 C0 C0
.:1080 00 00 03 0F 33 C3 FF 03
.:1088 03 00 00 00 00 00 C0 00
.:1090 00 00 FF C0 FF 00 00 C0
.:1098 3F 00 C0 00 00 C0 C0 C0
.:10A0 00 00 0F 30 C0 FF C0 C0
.:10A8 3F 00 C0 00 00 00 C0 C0
.:10B0 00 00 FF 00 03 0C 30 30
.:10B8 30 00 C0 C0 00 00 00 00
.:10C0 00 00 3F C0 C0 3F C0 C0
.:10C8 3F 00 00 C0 C0 00 C0 C0
.:10D0 00 00 3F C0 C0 3F 00 03
.:10D8 FC 00 00 C0 C0 C0 C0 00
.:10E0 00 00 00 00 00 00 00 00
.:10E8 00 00 00 00 00 00 00 00
.:10F0 00 00 00 28 50 78 A0 C8
.:10F8 F0 18 40 68 90 B8 E0 08
.:1100 30 58 80 A8 D0 F8 20 48
.:1108 70 98 C0 04 04 04 04 04
.:1110 04 04 05 05 05 05 05 05
.:1118 06 06 06 06 06 06 06 06
.:1120 07 07 07 07 07 00 01 02
.:1128 03 04 05 06 07 40 41 42
.:1130 43 44 45 46 47 80 81 82
.:1138 83 84 85 86 87 C0 C1 C2
.:1140 C3 C4 C5 C6 C7 00 01 02
.:1148 03 04 05 06 07 40 41 42
.:1150 43 44 45 46 47 80 81 82
.:1158 83 84 85 86 87 C0 C1 C2
.:1160 C3 C4 C5 C6 C7 00 01 02
.:1168 03 04 05 06 07 40 41 42
.:1170 43 44 45 46 47 80 81 82
.:1178 83 84 85 86 87 C0 C1 C2
.:1180 C3 C4 C5 C6 C7 00 01 02
.:1188 03 04 05 06 07 40 41 42
.:1190 43 44 45 46 47 80 81 82
.:1198 83 84 85 86 87 C0 C1 C2
.:11A0 C3 C4 C5 C6 C7 00 01 02
.:11A8 03 04 05 06 07 40 41 42
.:11B0 43 44 45 46 47 80 81 82
.:11B8 83 84 85 86 87 C0 C1 C2
.:11C0 C3 C4 C5 C6 C7 00 01 02
.:11C8 03 04 05 06 07 40 41 42
.:11D0 43 44 45 46 47 80 81 82
.:11D8 83 84 85 86 87 C0 C1 C2
.:11E0 C3 C4 C5 C6 C7 00 01 02
.:11E8 03 04 05 06 07 40 41 42
.:11F0 43 44 45 46 47 80 81 82
.:11F8 83 84 85 86 87 00 00 00
.:1200 00 00 00 00 00 01 01 01
.:1208 01 01 01 01 01 02 02 02
.:1210 02 02 02 02 02 03 03 03
.:1218 03 03 03 03 03 05 05 05
.:1220 05 05 05 05 05 06 06 06
.:1228 06 06 06 06 06 07 07 07
.:1230 07 07 07 07 07 08 08 08
.:1238 08 08 08 08 08 0A 0A 0A
.:1240 0A 0A 0A 0A 0A 0B 0B 0B
.:1248 0B 0B 0B 0B 0B 0C 0C 0C
.:1250 0C 0C 0C 0C 0C 0D 0D 0D
.:1258 0D 0D 0D 0D 0D 0F 0F 0F
.:1260 0F 0F 0F 0F 0F 10 10 10
.:1268 10 10 10 10 10 11 11 11
.:1270 11 11 11 11 11 12 12 12
.:1278 12 12 12 12 12 14 14 14
.:1280 14 14 14 14 14 15 15 15
.:1288 15 15 15 15 15 16 16 16
.:1290 16 16 16 16 16 17 17 17
.:1298 17 17 17 17 17 19 19 19
.:12A0 19 19 19 19 19 1A 1A 1A
.:12A8 1A 1A 1A 1A 1A 1B 1B 1B
.:12B0 1B 1B 1B 1B 1B 1C 1C 1C
.:12B8 1C 1C 1C 1C 1C 1E 1E 1E
.:12C0 1E 1E 1E 1E 1E 1F 1F 1F
.:12C8 1F 1F 1F 1F 1F 00 08 10
.:12D0 18 20 28 30 38 40 48 50
.:12D8 58 60 68 70 78 80 88 90
.:12E0 98 A0 A8 B0 B8 C0 C8 D0
.:12E8 D8 E0 E8 F0 F8 00 08 10
.:12F0 18 20 28 30 38 00 00 00
.:12F8 00 00 00 00 00 00 00 00
.:1300 00 00 00 00 00 00 00 00
.:1308 00 00 00 00 00 00 00 00
.:1310 00 00 00 00 00 01 01 01
.:1318 01 01 01 01 01 FF 00 72
.:1320 00 F1 00 FF 00 FF 00 FF
.:1328 00 FF 00 FF 00 FF 2F FF
.:1330 00 70 00 FF 00 FF 00 FF
.:1338 00 FF 00 FF 00 FF 4F F7
.:1340 00 72 00 DD 00 FF 00 58
.:1348 00 FF 00 EF 00 7D 00 FF
.:1350 00 FF 00 FF 00 FF 00 DF
.:1358 00 FF 00 FF 00 FF 00 F0
```

```
.:1360 00 7F 00 7D 20 FF 52 FF
.:1368 00 FF 00 FF 00 FF 00 FF
.:1370 00 FF 00 FF 00 FF 00 FF
.:1378 00 FF 00 FF 00 FF E0 F7
.:1380 00 DF 00 85 02 FF 00 DD
.:1388 00 FF 00 FF D0 FF 02 DF
.:1390 00 FF 00 FF 00 FF 00 FF
.:1398 00 FF 00 FF 00 FF 00 FF
.:13A0 00 FF 00 FF 7D DD 02 FF
.:13A8 00 FF 00 FF 00 FF D1 FF
.:13B0 00 7F 00 FF 00 D5 22 FF
.:13B8 00 FF 00 FF 00 FF 00 FF
.:13C0 05 FF 00 FF 00 FF 00 FD
.:13C8 00 FF 00 FF 00 FF 00 FF
.:13D0 00 FF 00 FF 00 FF 00 FF
.:13D8 00 FF 00 2F C0 FF 00 FF
.:13E0 00 FF 00 FF 00 FF 00 FF
.:13E8 DD FF 00 FF 00 FF 00 F6
.:13F0 4C C8 15 EE 5D 08 EE 5D
.:13F8 08 D0 39 AD 51 08 C9 FF
.:1400 4C 8F 18 01 02 04 08 10
.:1408 20 40 80 FE FD FB F7 EF
.:1410 DF BF 7F FE FD FB F7 EF
.:1418 DF BF 7F 00 04 08 0C 10
.:1420 14 04 04 06 02 0A 09 05
.:1428 04 04 06 06 04 05 05 01
.:1430 09 08 0A 02 06 06 06 04
.:1438 04 05 05 01 05 09 08 09
.:1440 08 0A 0A 02 06 06 04 06
.:1448 02 04 04 06 02 0A 09 05
.:1450 04 04 06 06 04 05 05 01
.:1458 09 08 0A 02 06 06 06 04
.:1460 04 05 05 01 05 09 08 09
.:1468 08 0A 0A 02 06 06 04 06
.:1470 02 00 00 00 04 06 06 06
.:1478 06 06 06 06 02 06 02 06
.:1480 02 0A 02 0A 0A 0A 08 0A
.:1488 08 0A 08 09 08 09 09 01
.:1490 09 01 05 01 05 05 01 01
.:1498 05 04 06 06 06 00 00 00
.:14A0 06 06 06 04 04 04 05 05
.:14A8 05 01 05 05 01 01 09 09
.:14B0 09 08 09 08 08 0A 08 0A
.:14B8 08 0A 0A 02 0A 02 02 06
.:14C0 06 02 06 06 04 06 04 00
.:14C8 00 00 05 05 08 08 0A 0A
.:14D0 02 02 06 06 04 04 04 06
.:14D8 06 02 0A 0A 0A 08 08 09
.:14E0 01 09 01 05 05 05 01 05
.:14E8 04 05 05 01 09 08 0A 02
.:14F0 06 02 05 05 08 08 0A 0A
.:14F8 02 02 06 06 04 04 04 06
.:1500 06 02 0A 0A 0A 08 08 09
.:1508 01 09 01 05 05 05 01 05
.:1510 04 05 05 01 09 08 0A 02
.:1518 06 02 00 00 00 04 06 04
.:1520 04 06 04 04 06 04 04 05
.:1528 04 04 05 05 04 05 05 05
.:1530 01 05 05 01 05 05 04 04
.:1538 06 06 02 06 02 06 06 04
.:1540 06 06 04 06 04 05 04 04
.:1548 06 06 02 06 02 06 06 04
.:1550 06 06 04 06 04 00 00 00
.:1558 04 06 04 06 04 06 04 06
.:1560 04 06 04 06 06 02 06 02
.:1568 06 02 06 02 05 01 05 01
.:1570 05 01 05 01 05 04 05 04
.:1578 05 04 05 04 05 04 05 04
.:1580 05 01 05 01 05 04 05 04
.:1588 05 04 05 04 00 00 00 04
.:1590 05 04 05 05 05 05 01 05
.:1598 01 05 01 05 01 05 01 05
.:15A0 01 05 01 04 06 06 02 06
.:15A8 02 02 06 02 02 06 04 05
.:15B0 05 04 06 06 04 05 05 00
.:15B8 00 00 04 05 05 01 01 05
.:15C0 01 05 04 06 02 06 02 06
.:15C8 06 04 04 05 05 01 01 06
.:15D0 06 04 06 06 02 02 06 04
.:15D8 05 05 05 04 04 06 06 06
.:15E0 02 06 00 00 00 04 05 04
.:15E8 06 04 05 04 05 01 05 01
.:15F0 04 01 05 04 05 02 05 04
.:15F8 06 04 06 02 06 04 06 02
.:1600 06 02 06 04 06 04 06 04
.:1608 05 01 05 04 06 00 00 00
.:1610 01 05 01 01 02 01 01 04
.:1618 01 05 08 01 05 04 06 04
.:1620 06 02 08 02 04 02 01 06
.:1628 02 06 04 02 05 01 04 08
.:1630 01 05 04 06 06 02 06 04
.:1638 00 00 00 02 05 02 06 02
.:1640 0A 09 0A 02 01 02 0A 04
.:1648 0A 02 08 01 08 02 09 08
.:1650 09 04 01 09 01 05 09 01
.:1658 04 05 01 04 06 05 04 01
.:1660 02 04 06 00 00 00 04 05
.:1668 04 01 05 02 05 01 09 05
.:1670 01 09 08 0A 09 05 0A 08
.:1678 0A 02 0A 08 02 06 04 01
.:1680 05 09 01 05 04 06 04 06
.:1688 02 08 02 06 04 05 00 00
.:1690 00 06 04 06 04 03 03 03
.:1698 06 02 06 02 06 02 06 03
.:16A0 03 03 03 05 05 05 05 05
.:16A8 03 03 03 03 04 04 04 03
.:16B0 03 03 05 01 01 05 03 03
.:16B8 03 00 00 00 04 06 04 04
.:16C0 06 04 04 06 03 03 03 03
.:16C8 05 05 05 05 05 03 03 03
.:16D0 03 05 01 01 05 01 01 05
.:16D8 01 01 03 03 03 04 04 04
.:16E0 03 03 03 03 03 00 00 00
.:16E8 01 01 01 03 03 03 03 03
.:16F0 0A 0A 0A 0A 0A 0A 0A 03
.:16F8 03 03 08 09 08 09 08 09
.:1700 08 09 03 03 03 0A 02 02
.:1708 0A 02 02 0A 03 03 03 03
```

```
.:1710 00 00 00 08 08 08 08 03
.:1718 03 03 09 09 09 03 03 03
.:1720 0A 0A 02 0A 0A 03 03 03
.:1728 03 02 02 02 02 03 03 03
.:1730 05 05 05 03 03 03 03 03
.:1738 00 00 00 06 06 06 06 06
.:1740 05 01 05 05 01 05 05 01
.:1748 05 06 06 05 05 05 05 02
.:1750 06 02 06 02 06 02 05 05
.:1758 02 06 06 01 05 01 05 01
.:1760 05 06 06 00 00 00 06 02
.:1768 06 02 05 05 05 06 06 06
.:1770 06 06 05 01 05 01 05 01
.:1778 06 06 05 05 01 05 06 02
.:1780 06 06 02 06 06 02 06 06
.:1788 02 05 05 05 00 00 00 21
.:1790 14 74 14 A0 14 CA 14 00
.:1798 00 00 00 1D 15 58 15 8F
.:17A0 15 BA 15 00 00 00 00 E5
.:17A8 15 10 16 3B 16 66 16 00
.:17B0 00 00 00 91 16 BC 16 E8
.:17B8 16 13 17 00 00 00 00 3B
.:17C0 17 66 17 66 17 3B 17 00
.:17C8 00 00 00 8F 17 9B 17 A7
.:17D0 17 B3 17 BF 17 00 01 02
.:17D8 03 04 01 00 04 03 02 01
.:17E0 03 02 00 04 03 02 01 00
.:17E8 00 42 10 52 10 62 10 72
.:17F0 10 82 10 92 10 A2 10 B2
.:17F8 10 C2 10 D2 10 E2 10 00
.:1800 00 01 01 01 01 01 01 02
.:1808 02 02 03 03 04 03 04 03
.:1810 05 06 01 02 01 01 01 01
.:1818 02 03 02 02 03 05 02 05
.:1820 04 06 07 01 02 04 04 05
.:1828 08 08 06 06 05 08 06 09
.:1830 06 06 08 09 04 06 05 04
.:1838 00 55 AA FF C0 30 0C 03
.:1840 A5 66 38 E9 32 C9 C7 B0
.:1848 44 AA A5 67 38 E9 0E C9
.:1850 A0 B0 3A 4A 4A A8 A5 64
.:1858 18 7D 25 11 85 61 A5 65
.:1860 7D FD 11 85 62 A5 61 18
.:1868 79 CD 12 85 61 A5 62 79
.:1870 F5 12 85 62 A5 67 29 03
.:1878 AA A0 00 B1 61 85 63 A4
.:1880 60 B9 38 18 3D 3C 18 45
.:1888 63 A0 00 91 61 60 40 78
.:1890 78 A9 06 85 01 A9 00 8D
.:1898 0D DC 8D 0D DD AD 0D DC
.:18A0 AD 0D DD A9 00 8D 02 DC
.:18A8 8D 03 DC 8D 0E DC 8D 0F
.:18B0 DC 8D 0E DD 8D 0F DD A9
.:18B8 01 8D 1A D0 A9 FF 8D 19
.:18C0 D0 A2 02 A9 00 95 00 E8
.:18C8 D0 FB A2 FF 9A A9 8E 8D
.:18D0 18 03 A9 18 8D 19 03 A9
.:18D8 BD 8D 14 03 A9 1F 8D 15
.:18E0 03 A9 00 8D 20 D0 8D 21
.:18E8 D0 A9 03 8D 02 DD 38 E9
.:18F0 01 8D 00 DD A9 20 4A 4A
.:18F8 85 44 AD 18 D0 29 F0 05
.:1900 44 8D 18 D0 A9 1C 0A 0A
.:1908 85 44 AD 18 D0 29 0F 05
.:1910 44 8D 18 D0 A9 00 85 64
.:1918 A9 60 85 65 AD 11 D0 09
.:1920 20 8D 11 D0 AD 16 D0 09
.:1928 10 8D 16 D0 A9 00 85 24
.:1930 A9 D8 85 25 A0 00 A2 00
.:1938 A9 01 91 24 C8 D0 FB E6
.:1940 25 A5 25 C9 DC D0 F1 A9
.:1948 00 85 24 A9 60 85 25 A0
.:1950 00 A2 00 A9 00 91 24 C8
.:1958 D0 FB E6 25 E8 E0 20 D0
.:1960 F2 A9 00 85 24 A9 5C 85
.:1968 25 A0 00 A2 00 A9 56 91
.:1970 24 C8 D0 FB E6 25 E8 E0
.:1978 04 D0 F2 A9 03 85 6B 85
.:1980 60 A9 01 85 69 A9 01 85
.:1988 6A A9 14 85 BC A9 2A 85
.:1990 70 85 71 85 72 85 73 A9
.:1998 2D 85 74 85 75 85 76 85
.:19A0 77 A9 00 85 78 A9 30 85
.:19A8 79 A9 60 85 7A A9 90 85
.:19B0 7B A9 C0 85 7C A9 F0 85
.:19B8 7D A9 20 85 7E A9 50 85
.:19C0 7F A9 09 85 88 85 89 85
.:19C8 8A 85 8B 85 8C 85 8D 85
.:19D0 8E 85 8F A9 20 85 80 A9
.:19D8 21 85 81 A9 22 85 82 A9
.:19E0 23 85 83 A9 20 85 84 A9
.:19E8 21 85 85 A9 22 85 86 A9
.:19F0 23 85 87 A9 01 85 A4 A9
.:19F8 05 85 A5 85 A6 85 A7 85
.:1A00 A8 85 A9 85 AA 85 AB A9
.:1A08 02 8D 25 D0 A9 07 8D 26
.:1A10 D0 A9 FF 8D 1B D0 A9 C0
.:1A18 85 90 A9 00 85 6C 85 6D
.:1A20 85 6E 85 6F A9 01 85 37
.:1A28 A9 18 85 07 A9 0E 85 08
.:1A30 A9 C3 85 05 A9 10 85 06
.:1A38 A9 0F 85 19 A9 20 85 02
.:1A40 A9 28 85 11 A9 88 85 14
.:1A48 A9 79 85 09 A9 64 85 0A
.:1A50 A9 88 8D 12 D4 A9 80 8D
.:1A58 12 D4 85 04 A9 FF 85 91
.:1A60 A9 30 85 94 A9 70 85 9C
.:1A68 A9 68 85 9D A9 70 85 9E
.:1A70 A9 80 85 9F A9 90 85 A0
.:1A78 A9 A0 85 A1 A9 B0 85 A2
.:1A80 A9 C5 85 A3 A9 90 85 95
.:1A88 A9 95 85 96 A9 85 85 97
.:1A90 A9 90 85 98 A9 70 85 99
.:1A98 A9 75 85 9A A9 95 85 9B
.:1AA0 A9 21 85 C4 A9 14 85 C5
.:1AA8 A9 74 85 C6 A9 14 85 C7
.:1AB0 A9 A0 85 C8 A9 14 85 C9
.:1AB8 A9 CA 85 CA A9 14 85 CB
```

```
.:1AC0 A9 21 85 CC A9 14 85 CD
.:1AC8 A9 74 85 CE A9 14 85 CF
.:1AD0 A9 A0 85 D0 A9 14 85 D1
.:1AD8 AD 1E D0 AD 1F D0 A9 FB
.:1AE0 8D 12 D0 AD 11 D0 29 7F
.:1AE8 8D 11 D0 58 A5 6B 05 B4
.:1AF0 05 B5 05 B6 05 B7 05 B8
.:1AF8 05 B9 05 BA 05 BB D0 03
.:1B00 4C A8 1F A5 38 29 0F C9
.:1B08 0F F0 03 4C D1 1B A9 0A
.:1B10 85 3C 85 3D 85 3E 85 3F
.:1B18 85 40 85 41 85 42 85 43
.:1B20 A5 6F 29 F0 4A 4A 4A 4A
.:1B28 85 3C A5 6F 29 0F 85 3D
.:1B30 A5 6E 29 F0 4A 4A 4A 4A
.:1B38 85 3E A5 6E 29 0F 85 3F
.:1B40 A5 6D 29 F0 4A 4A 4A 4A
.:1B48 85 40 A5 6D 29 0F 85 41
.:1B50 A5 6C 29 F0 4A 4A 4A 4A
.:1B58 85 42 A5 6C 29 0F 85 43
.:1B60 A5 3C D0 34 A9 0A 85 3C
.:1B68 A5 3D D0 2C A9 0A 85 3D
.:1B70 A5 3E D0 24 A9 0A 85 3E
.:1B78 A5 3F D0 1C A9 0A 85 3F
.:1B80 A5 40 D0 14 A9 0A 85 40
.:1B88 A5 41 D0 0C A9 0A 85 41
.:1B90 A5 42 D0 04 A9 0A 85 42
.:1B98 A2 00 A9 00 85 5C A9 00
.:1BA0 85 46 A9 60 85 47 B5 3C
.:1BA8 0A A8 B9 E9 17 85 44 B9
.:1BB0 EA 17 85 45 A0 00 B1 44
.:1BB8 91 46 C8 C0 10 D0 F7 A5
.:1BC0 46 18 69 10 85 46 A9 00
.:1BC8 65 47 85 47 E8 E0 08 D0
.:1BD0 D5 A5 38 29 01 D0 03 4C
.:1BD8 79 1C A2 00 A0 00 84 5C
.:1BE0 A5 91 A4 5C 39 03 14 F0
.:1BE8 76 B9 B5 00 D0 71 A1 C4
.:1BF0 D0 32 A4 37 B9 D5 17 0A
.:1BF8 A8 B9 CB 17 85 3C B9 CC
.:1C00 17 85 3D A5 33 29 06 A8
.:1C08 B1 3C 95 C4 C8 B1 3C 95
.:1C10 C5 8A 65 33 45 38 29 07
.:1C18 75 C4 95 C4 A9 00 75 C5
.:1C20 95 C5 A1 C4 85 44 A4 37
.:1C28 B9 01 18 85 69 B9 12 18
.:1C30 85 6A A4 5C B9 AD 00 F0
.:1C38 06 A5 44 49 FF 85 44 A5
.:1C40 44 85 45 66 44 90 03 20
.:1C48 A2 22 66 44 90 03 20 B0
.:1C50 22 66 44 90 03 20 BE 22
.:1C58 66 44 90 03 20 CC 22 B5
.:1C60 C4 18 69 01 95 C4 B5 C5
.:1C68 69 00 95 C5 E8 E8 E6 5C
.:1C70 A5 5C C9 07 F0 03 4C E0
.:1C78 1B A5 91 29 7F F0 03 4C
.:1C80 49 1D A9 FF 85 91 A2 00
.:1C88 A9 70 85 3C A5 33 29 7F
.:1C90 65 3C 85 9D A5 34 29 7F
.:1C98 65 3C 85 9E A5 35 29 7F
.:1CA0 65 3C 85 9F A5 36 69 7F
.:1CA8 65 3C 85 A0 A5 33 45 34
.:1CB0 29 7F 65 3C 85 A1 A5 34
.:1CB8 45 35 29 7F 65 3C 85 A2
.:1CC0 A5 35 45 33 29 7F 65 3C
.:1CC8 85 A3 A9 90 85 95 A9 85
.:1CD0 85 96 A9 95 85 97 A9 70
.:1CD8 85 98 A9 75 85 99 A9 90
.:1CE0 85 9A A9 80 85 9B A9 00
.:1CE8 85 AD 85 AE 85 AF 85 B0
.:1CF0 85 B1 85 B2 85 B3 A9 E2
.:1CF8 85 C4 A9 10 85 C5 A9 E2
.:1D00 85 C6 A9 10 85 C7 A9 E2
.:1D08 85 C8 A9 10 85 C9 A9 E2
.:1D10 85 CA A9 10 85 CB A9 E2
.:1D18 85 CC A9 10 85 CD A9 E2
.:1D20 85 CE A9 10 85 CF A6 37
.:1D28 BD D5 17 0A 0A AA 86 BD
.:1D30 86 C3 E8 86 BE 86 C0 E8
.:1D38 86 BF E8 86 C1 86 C2 E6
.:1D40 37 A5 37 C9 12 90 02 C6
.:1D48 37 A2 00 A0 00 B5 95 C9
.:1D50 16 B0 04 A9 01 95 AD C9
.:1D58 98 90 04 A9 00 95 AD E8
.:1D60 E0 07 D0 E9 A5 38 29 01
.:1D68 F0 03 4C F9 1D A5 92 29
.:1D70 01 D0 03 4C F9 1D A5 91
.:1D78 0A 85 3C A5 92 29 FE 25
.:1D80 3C F0 76 4A 85 3C A2 00
.:1D88 66 3C 90 68 B5 B5 D0 64
.:1D90 B5 9D 18 69 10 38 E5 DD
.:1D98 C9 10 B0 58 B5 95 18 69
.:1DA0 0C 38 E5 D5 C9 0C B0 4C
.:1DA8 A9 C0 95 B5 A9 18 95 BD
.:1DB0 A9 79 85 07 A9 64 85 08
.:1DB8 A9 05 85 12 A9 00 85 15
.:1DC0 A9 80 8D 0B D4 A9 81 85
.:1DC8 03 A9 18 85 1C A5 37 0A
.:1DD0 0A 0A F8 18 65 6C 85 6C
.:1DD8 A5 6D 69 01 85 6D A5 6E
.:1DE0 69 00 85 6E A5 6F 69 00
.:1DE8 85 6F D8 A5 D4 29 FE 85
.:1DF0 D4 4C F9 1D E8 E0 07 D0
.:1DF8 8F A5 38 29 01 F0 31 A5
.:1E00 93 29 01 F0 2B A5 6B F0
.:1E08 27 A5 B4 D0 23 C6 6B A9
.:1E10 C0 85 B4 A9 18 85 BC A9
.:1E18 30 85 1B A9 80 8D 04 D4
.:1E20 A9 1A 85 11 A9 81 85 02
.:1E28 A9 3E 85 05 A9 2A 85 06
.:1E30 A5 38 29 03 D0 70 A2 00
.:1E38 A5 D4 49 FF 4A 25 91 85
.:1E40 3C 66 3C 90 5C A4 37 B9
.:1E48 23 18 C5 34 90 58 B5 B5
.:1E50 D0 4F A5 D4 1D 04 14 85
.:1E58 D4 B5 9D 18 69 08 95 DE
.:1E60 B5 95 95 D6 A5 35 29 03
.:1E68 A8 B9 34 18 95 E6 86 44
```

```
.:1E70 B5 D6 85 67 B5 DE 85 66
.:1E78 20 40 18 A9 1F 85 07 A9
.:1E80 15 85 08 8A 18 65 08 85
.:1E88 08 A9 20 8D 0B D4 A9 09
.:1E90 85 12 A9 00 85 15 A9 21
.:1E98 85 03 A9 14 85 1C 4C A6
.:1EA0 1E E8 E0 07 D0 9B A5 38
.:1EA8 29 01 D0 03 4C 10 1F A9
.:1EB0 03 85 60 A2 01 86 5C A5
.:1EB8 D4 85 3C 66 3C A6 5C 66
.:1EC0 3C 90 42 B5 D5 85 67 B5
.:1EC8 DD 85 66 20 40 18 A6 5C
.:1ED0 A4 37 B5 E5 85 44 66 44
.:1ED8 90 08 B5 DD 18 F9 12 18
.:1EE0 95 DD 66 44 90 08 B5 DD
.:1EE8 38 79 12 18 95 DD 66 44
.:1EF0 90 08 B5 D5 18 F9 01 18
.:1EF8 95 D5 B5 D5 85 67 B5 DD
.:1F00 85 66 20 40 18 E6 5C A5
.:1F08 5C C9 08 F0 03 4C BD 1E
.:1F10 A5 D4 29 01 F0 04 E6 D5
.:1F18 E6 D5 A5 D4 29 01 F0 0C
.:1F20 A5 D5 C9 A0 90 06 A5 D4
.:1F28 29 FE 85 D4 A2 01 B5 D5
.:1F30 C9 0C B0 03 4C 44 1F B5
.:1F38 DD C9 F4 90 03 4C 44 1F
.:1F40 C9 50 B0 16 A5 D4 3D 13
.:1F48 14 85 D4 86 3C B5 D5 85
.:1F50 67 B5 DD 85 66 20 40 18
.:1F58 A6 3C E8 E0 08 D0 CF A5
.:1F60 38 29 07 C9 06 D0 1A A2
.:1F68 00 B5 BC 29 03 A8 B5 BC
.:1F70 29 FC 85 3C C8 98 29 03
.:1F78 05 3C 95 BC E8 E0 08 D0
.:1F80 E8 A5 38 29 0F D0 21 A9
.:1F88 C0 85 3C A9 60 85 3D A5
.:1F90 6B 0A A8 B9 E9 17 85 3E
.:1F98 B9 EA 17 85 3F A0 00 B1
.:1FA0 3E 91 3C C8 C0 10 D0 F7
.:1FA8 AD 01 DC 29 10 D0 03 4C
.:1FB0 8F 18 A9 00 85 3B A5 3B
.:1FB8 F0 FC 4C EC 1A AD 1E D0
.:1FC0 85 92 AD 1F D0 85 93 A9
.:1FC8 06 8D 21 D0 E6 38 E6 39
.:1FD0 A5 39 49 3C D0 04 85 39
.:1FD8 E6 3A A2 00 86 68 A0 00
.:1FE0 B5 70 99 01 D0 B5 80 9D
.:1FE8 F8 5F B5 78 99 00 D0 B5
.:1FF0 88 9D 27 D0 C8 C8 E8 E0
.:1FF8 08 D0 E5 A5 90 8D 10 D0
.:2000 A9 00 8D 1C D0 A9 FF 8D
.:2008 1D D0 8D 17 D0 8D 15 D0
.:2010 A5 B4 F0 0A C6 B4 A5 B4
.:2018 D0 04 A9 14 85 BC A5 1B
.:2020 F0 0A C6 1B D0 06 A5 02
.:2028 29 FE 85 02 A5 1C F0 0A
.:2030 C6 1C D0 06 A5 03 29 FE
.:2038 85 03 A5 1D F0 04 C6 1D
.:2040 D0 00 A5 11 8D 05 D4 A5
.:2048 12 8D 0C D4 A5 13 8D 13
.:2050 D4 A5 14 8D 06 D4 A5 15
.:2058 8D 0D D4 A5 16 8D 14 D4
.:2060 A5 05 8D 00 D4 A5 07 8D
.:2068 07 D4 A5 09 8D 0E D4 A5
.:2070 06 8D 01 D4 A5 08 8D 08
.:2078 D4 A5 0A 8D 0F D4 A5 0B
.:2080 8D 02 D4 A5 0D 8D 09 D4
.:2088 A5 0F 8D 10 D4 A5 0C 8D
.:2090 03 D4 A5 0E 8D 0A D4 A5
.:2098 10 8D 11 D4 A5 17 8D 15
.:20A0 D4 A5 18 8D 16 D4 A5 19
.:20A8 8D 18 D4 A5 1A 8D 17 D4
.:20B0 A5 02 8D 04 D4 A5 03 8D
.:20B8 0B D4 A5 04 8D 12 D4 A5
.:20C0 38 29 03 AA AD 1B D4 65
.:20C8 33 65 35 45 34 E5 36 65
.:20D0 39 95 33 A5 6B D0 03 4C
.:20D8 6F 21 A5 B4 F0 03 4C 6F
.:20E0 21 AD 00 DC 49 FF 85 4D
.:20E8 29 0F F0 46 A5 38 29 01
.:20F0 D0 40 A5 4D 85 4C 66 4C
.:20F8 90 0B A5 9C 38 E9 01 C9
.:2100 68 90 02 85 9C 66 4C 90
.:2108 0B A5 9C 18 69 01 C9 E8
.:2110 B0 02 85 9C 66 4C 90 0B
.:2118 A5 94 38 E9 01 C9 10 90
.:2120 02 85 94 66 4C 90 0B A5
.:2128 94 18 69 01 C9 30 B0 02
.:2130 85 94 AD 00 DC 29 10 D0
.:2138 36 A5 D4 29 01 D0 30 A5
.:2140 B4 D0 2C A5 D4 09 01 85
.:2148 D4 A5 94 18 69 05 85 D5
.:2150 A5 9C 18 69 07 85 DD A9
.:2158 08 85 E5 A9 81 85 02 A9
.:2160 20 85 1B A9 61 85 05 A9
.:2168 08 85 06 A9 28 85 11 A9
.:2170 95 8D 14 03 A9 21 8D 15
.:2178 03 A9 01 85 3B A9 54 8D
.:2180 12 D0 AD 11 D0 29 7F 8D
.:2188 11 D0 A9 FF 8D 19 D0 68
.:2190 A8 68 AA 68 40 78 AD 1E
.:2198 D0 AD 1F D0 EA EA EA A9
.:21A0 05 8D 21 D0 A2 00 86 68
.:21A8 A0 00 B5 9C 99 01 D0 B5
.:21B0 BC 9D F8 5F B5 94 0A 99
.:21B8 00 D0 90 07 A5 68 1D 03
.:21C0 14 85 68 C8 C8 E8 E0 08
.:21C8 D0 E0 A5 68 8D 10 D0 A5
.:21D0 38 29 01 F0 25 A5 D4 29
.:21D8 01 F0 1F A9 1C 8D F8 5F
.:21E0 A5 DD 8D 01 D0 A5 D5 0A
.:21E8 8D 00 D0 AD 10 D0 29 FE
.:21F0 8D 10 D0 90 05 09 01 8D
.:21F8 10 D0 A9 00 8D 1D D0 8D
.:2200 17 D0 A9 FF 8D 1C D0 A5
.:2208 91 0A 09 01 8D 15 D0 A2
.:2210 00 A5 90 85 68 D6 78 B5
.:2218 78 C9 FF D0 17 A5 68 3D
```

```
.:2220 03 14 D0 09 A5 68 1D 03
.:2228 14 85 68 D0 07 A5 68 3D
.:2230 0B 14 85 68 E8 E0 08 D0
.:2238 DC A5 68 85 90 A2 00 A5
.:2240 90 3D 03 14 F0 02 A9 01
.:2248 85 4D B5 78 85 4C A5 4D
.:2250 C9 01 D0 11 A5 4C C9 E8
.:2258 D0 0B A5 90 1D 03 14 85
.:2260 90 A9 70 95 78 E8 E0 08
.:2268 D0 D5 A2 00 B5 B5 F0 0B
.:2270 D6 B5 D0 07 A5 91 3D 13
.:2278 14 85 91 E8 E0 07 D0 EC
.:2280 A9 BD 8D 14 03 A9 1F 8D
.:2288 15 03 A9 FB 8D 12 D0 AD
.:2290 11 D0 29 7F 8D 11 D0 A9
.:2298 FF 8D 19 D0 68 A8 68 AA
.:22A0 68 40 B9 9D 00 38 E5 6A
.:22A8 C9 68 90 03 99 9D 00 60
.:22B0 B9 9D 00 18 65 6A C9 E8
.:22B8 B0 03 99 9D 00 60 B9 95
.:22C0 00 38 E5 69 C9 0E 90 03
.:22C8 99 95 00 60 B9 95 00 18
.:22D0 65 69 C9 A9 B0 03 99 95
.:22D8 00 60 EA 00 FF FF 00 00
.:22E0 FF FF 00 00 FF FF 00 00
.
.
.:4000 20 00 00 8C 00 00 82 00
.:4008 20 80 FA 80 3A AE 00 00
.:4010 EA C0 0A 08 20 0E 08 0C
.:4018 00 02 A0 00 00 00 00 00
.:4020 00 00 00 00 00 00 00 00
.:4028 00 00 00 00 00 00 00 00
.:4030 00 00 00 00 00 00 00 00
.:4038 00 00 00 00 00 00 00 00
.:4040 2C 00 00 82 00 00 82 00
.:4048 20 20 80 80 20 8E 00 0A
.:4050 AB A8 00 EA 02 0E B8 02
.:4058 0A 02 BC 00 00 00 00 00
.:4060 00 00 00 00 00 00 00 00
.:4068 00 00 00 00 00 00 00 00
.:4070 00 00 00 00 00 00 00 00
.:4078 00 00 00 00 00 00 00 00
.:4080 20 00 00 8F 00 20 80 80
.:4088 80 CA 8F 00 20 FA 00 00
.:4090 AA C0 0A A0 08 0B 08 02
.:4098 00 03 C2 00 00 28 00 00
.:40A0 00 00 00 00 00 00 00 00
.:40A8 00 00 00 00 00 00 00 00
.:40B0 00 00 00 00 00 00 00 00
.:40B8 00 00 00 00 00 00 00 00
.:40C0 00 00 20 0A 8E 80 30 AB
.:40C8 00 80 EB C0 C0 B8 20 2B
.:40D0 0C 08 0A 02 08 00 00 F0
.:40D8 00 00 00 00 00 00 00 00
.:40E0 00 00 00 00 00 00 00 00
.:40E8 00 00 00 00 00 00 00 00
.:40F0 00 00 00 00 00 00 00 00
.:40F8 00 00 00 00 00 00 00 00
.:4100 01 00 80 0A 02 60 26 49
.:4108 A4 60 AA 0A 90 18 09 00
.:4110 20 00 02 48 00 02 08 00
.:4118 00 00 00 00 00 00 00 00
.:4120 00 00 00 00 00 00 00 00
.:4128 00 00 00 00 00 00 00 00
.:4130 00 00 00 00 00 00 00 00
.:4138 00 00 00 00 00 00 00 00
.:4140 09 00 60 1A 82 98 A0 41
.:4148 04 40 8A 0A 00 24 01 00
.:4150 28 02 00 60 00 02 A0 00
.:4158 01 20 00 00 18 00 00 80
.:4160 00 00 00 00 00 00 00 00
.:4168 00 00 00 00 00 00 00 00
.:4170 00 00 00 00 00 00 00 00
.:4178 00 00 00 00 00 00 00 00
.:4180 80 00 06 10 00 28 24 00
.:4188 90 02 01 A0 06 0A 40 01
.:4190 26 00 02 68 00 00 90 00
.:4198 00 80 00 01 00 00 0A 40
.:41A0 00 20 80 00 02 20 00 01
.:41A8 08 00 00 00 00 00 00 00
.:41B0 00 00 00 00 00 00 00 00
.:41B8 00 00 00 00 00 00 00 00
.:41C0 00 00 60 00 02 A4 06 01
.:41C8 18 29 8A 08 18 98 02 A0
.:41D0 64 01 40 20 00 00 10 00
.:41D8 00 60 00 06 A0 00 08 84
.:41E0 00 00 48 00 00 00 00 00
.:41E8 00 00 00 00 00 00 00 00
.:41F0 00 00 00 00 00 00 00 00
.:41F8 00 00 00 00 00 00 00 00
.:4200 0A 82 A0 A0 28 0A 00 A0
.:4208 00 00 80 00 00 00 00 00
.:4210 00 00 00 00 00 00 00 00
.:4218 00 00 00 00 00 00 00 00
.:4220 00 00 00 00 00 00 00 00
.:4228 00 00 00 00 00 00 00 00
.:4230 00 00 00 00 00 00 00 00
.:4238 00 00 00 00 00 00 00 00
.:4240 0A 00 A0 A0 82 08 00 28
.:4248 02 00 A0 00 00 20 00 00
.:4250 00 00 00 00 00 00 00 00
.:4258 00 00 00 00 00 00 00 00
.:4260 00 00 00 00 00 00 00 00
.:4268 00 00 00 00 00 00 00 00
.:4270 00 00 00 00 00 00 00 00
.:4278 00 00 00 00 00 00 00 00
.:4280 A0 00 0A 08 00 A0 02 0A
.:4288 00 00 A0 00 02 80 00 00
.:4290 80 00 00 00 00 00 00 00
.:4298 00 00 00 00 00 00 00 00
.:42A0 00 00 00 00 00 00 00 00
.:42A8 00 00 00 00 00 00 00 00
.:42B0 00 00 00 00 00 00 00 00
.:42B8 00 00 00 00 00 00 00 00
.:42C0 00 00 A0 0A 02 08 20 A8
.:42C8 02 80 20 00 00 A0 00 00
.:42D0 00 00 00 00 00 00 00 00
```

```
.:42D8 00 00 00 00 00 00 00 00
.:42E0 00 00 00 00 00 00 00 00
.:42E8 00 00 00 00 00 00 00 00
.:42F0 00 00 00 00 00 00 00 00
.:42F8 00 00 00 00 00 00 00 00
.:4300 08 00 00 0C 00 00 2E 28
.:4308 00 2E A8 00 2A AB 00 AA
.:4310 AB 00 AA AB C0 AA AA 80
.:4318 2A AA 80 02 AA 80 03 EA
.:4320 80 02 EA 80 02 EF 80 02
.:4328 0F 00 02 0B 00 02 0B C0
.:4330 02 02 80 02 0A 80 0A 28
.:4338 00 00 00 00 00 00 00 00
.:4340 08 00 00 08 00 00 2A 00
.:4348 00 2E 00 00 2A 80 00 AE
.:4350 A0 00 AA AC 00 AA AC 00
.:4358 2A AF 00 03 AA 00 03 EA
.:4360 00 02 EA 00 0A EA 00 08
.:4368 3E 00 20 0C 00 20 0C 00
.:4370 80 0E 00 00 02 00 00 02
.:4378 00 00 08 00 00 08 00 00
.:4380 02 00 00 02 0B 00 0B 7B
.:4388 00 2B AB C0 2A AA 80 2A
.:4390 AA 80 0A EA 80 02 FA C0
.:4398 00 BA C0 0A B8 F0 20 00
.:43A0 30 80 00 08 00 00 02 00
.:43A8 00 00 00 00 00 00 00 00
.:43B0 00 00 00 00 00 00 00 00
.:43B8 00 00 00 00 00 00 00 00
.:43C0 08 00 00 0A 2C 00 2A AC
.:43C8 00 2E AF 00 AA AA 00 AA
.:43D0 AA 80 AB AA 80 2B EA 80
.:43D8 02 EA 80 02 EB 00 02 03
.:43E0 00 02 03 00 08 03 80 08
.:43E8 00 80 08 00 80 20 02 00
.:43F0 00 00 00 00 00 00 00 00
.:43F8 00 00 00 00 00 00 00 00
.:4400 28 00 00 2A 00 00 42 00
.:4408 00 03 00 00 02 00 00 02
.:4410 00 00 01 00 00 0A 00 00
.:4418 09 20 00 08 20 00 18 AC
.:4420 00 20 80 40 30 80 80 23
.:4428 88 00 22 02 00 31 02 00
.:4430 23 2C 00 22 20 20 21 20
.:4438 C0 2E 20 80 08 0A 00 00
.:4440 28 00 00 AD 00 00 82 00
.:4448 00 C1 00 00 02 00 00 02
.:4450 00 00 03 00 00 02 00 00
.:4458 02 02 00 02 03 80 01 08
.:4460 80 0A 0C 80 08 08 70 08
.:4468 28 80 0C 28 80 08 30 20
.:4470 08 20 10 08 20 20 04 10
.:4478 20 0A 80 12 02 80 04 00
.:4480 08 00 00 3E 00 00 22 00
.:4488 00 B1 80 00 80 80 00 C0
.:4490 80 00 00 40 00 00 80 00
.:4498 00 80 00 00 C0 00 00 80
.:44A0 00 00 70 20 00 20 88 00
.:44A8 10 8C 00 20 C8 00 20 81
.:44B0 00 22 02 00 32 02 00 23
.:44B8 02 00 2A 03 00 04 00 00
.:44C0 38 00 00 A8 00 00 24 00
.:44C8 00 08 00 00 08 00 00 0C
.:44D0 00 00 08 00 00 08 00 00
.:44D8 02 00 00 02 02 00 03 02
.:44E0 00 02 04 80 01 08 80 02
.:44E8 08 C0 02 0C 80 02 08 80
.:44F0 03 08 41 02 28 22 02 10
.:44F8 22 02 A0 23 00 80 04 00
.:4500 00 50 00 00 70 00 00 70
.:4508 00 00 C0 00 00 C0 00 02
.:4510 80 00 0E 80 00 32 B0 00
.:4518 32 8C 00 32 83 C0 32 80
.:4520 00 02 80 00 02 80 00 02
.:4528 20 00 08 08 00 08 08 00
.:4530 20 08 00 20 08 00 80 08
.:4538 00 20 0A 00 00 00 00 00
.:4540 00 50 00 00 50 00 00 70
.:4548 00 00 70 00 00 C0 00 02
.:4550 80 00 03 B0 00 0E B0 00
.:4558 32 8C 00 32 8C 00 32 83
.:4560 00 32 80 00 02 80 00 02
.:4568 20 00 02 20 00 02 08 00
.:4570 08 08 00 A0 08 00 80 08
.:4578 00 0A 00 00 00 00 00 00
.:4580 00 50 00 00 70 00 00 70
.:4588 00 00 C0 00 00 C0 00 00
.:4590 80 00 02 80 00 03 80 00
.:4598 0E 80 00 0E 80 00 0F 80
.:45A0 00 02 BC 00 02 80 00 02
.:45A8 80 00 02 80 00 02 20 00
.:45B0 08 20 00 20 20 00 20 20
.:45B8 00 00 28 00 00 00 00 00
.:45C0 00 50 00 00 70 00 00 70
.:45C8 00 00 C0 00 00 C0 00 00
.:45D0 80 00 03 80 00 0D B0 00
.:45D8 32 8C 00 32 83 C0 32 80
.:45E0 00 32 80 00 02 80 00 02
.:45E8 80 00 00 80 00 00 A0 00
.:45F0 02 20 00 02 08 00 08 08
.:45F8 80 08 02 00 02 00 00 00
.:4600 00 00 00 00 00 00 00 00
.:4608 00 00 00 00 03 00 00 03
.:4610 F0 00 0F 70 00 0D 7C 00
.:4618 03 5C 00 03 FC 00 00 C0
.:4620 00 00 00 00 00 00 00 00
.:4628 00 00 00 00 00 00 00 00
.:4630 00 00 00 00 00 00 00 00
.:4638 00 00 00 00 00 00 00 00
.:4640 00 00 00 00 00 00 02 A0
.:4648 00 0A C8 00 23 FA 00 2D
.:4650 5C 80 2D 5E 80 BD 1F 80
.:4658 2C 57 80 8D 7C 80 2F F3
.:4660 80 0B AA 00 02 80 00 00
.:4668 00 00 00 00 00 00 00 00
.:4670 00 00 00 00 00 00 00 00
.:4678 00 00 00 00 00 00 00 00
.:4680 02 AA 00 02 FE 80 0A F3
```

```
.:4688 80 2B 9E A0 2F 5D 80 29
.:4690 14 80 BD 57 E0 B5 45 E0
.:4698 8D 55 38 2F 17 E0 23 D3
.:46A0 80 2A DE 00 0A DF 80 DB
.:46A8 FE 00 0A E8 00 02 A0 00
.:46B0 00 80 00 00 00 00 00 00
.:46B8 00 00 00 00 00 00 00 00
.:46C0 02 A8 00 02 2A 80 0A AA
.:46C8 A0 2A AA A0 AA FA 80 2A
.:46D0 FE A0 A3 FF A0 2B D2 A0
.:46D8 2A DE A0 2A F6 A0 0A 9F
.:46E0 A0 2B 3E 80 2B CF 00 22
.:46E8 AB A0 2A AA A0 0A AB A0
.:46F0 0A 2A A0 00 8A A0 00 AA
.:46F8 80 00 2A 80 00 00 00 00
.:4700 18 00 00 A5 00 00 28 00
.:4708 00 00 00 00 00 00 00 00
.:4710 00 00 00 00 00 00 00 00
.:4718 00 00 00 00 00 00 00 00
.:4720 00 00 00 00 00 00 00 00
.:4728 00 00 00 00 00 00 00 00
.:4730 00 00 00 00 00 00 00 00
.:4738 00 00 00 00 00 00 00 00
.:4740 28 00 00 6A 00 00 18 00
.:4748 00 00 00 00 00 00 00 00
.:4750 00 00 00 00 00 00 00 00
.:4758 00 00 00 00 00 00 00 00
.:4760 00 00 00 00 00 00 00 00
.:4768 00 00 00 00 00 00 00 00
.:4770 00 00 00 00 00 00 00 00
.:4778 00 00 00 00 00 00 00 00
.:4780 27 00 00 5A 00 00 28 00
.:4788 00 00 00 00 00 00 00 00
.:4790 00 00 00 00 00 00 00 00
.:4798 00 00 00 00 00 00 00 00
.:47A0 00 00 00 00 00 00 00 00
.:47A8 00 00 00 00 00 00 00 00
.:47B0 00 00 00 00 00 00 00 00
.:47B8 00 00 00 00 00 00 00 00
.:47C0 28 00 00 A5 00 00 27 00
.:47C8 00 00 00 00 00 00 00 00
.:47D0 00 00 00 00 00 00 00 00
.:47D8 00 00 00 00 00 00 00 00
.:47E0 00 00 00 00 00 00 00 00
.:47E8 00 00 00 00 00 00 00 00
.:47F0 00 00 00 00 00 00 00 00
.:47F8 00 00 00 00 00 00 00 00
.:4800 00 00 01 00 00 03 00 00
.:4808 07 00 00 0F 00 00 1F 00
.:4810 00 1F 00 00 7F 00 03 7E
.:4818 00 07 FD 00 6F BB 00 FF
.:4820 B7 01 FF EF 03 FF F7 03
.:4828 FF FB 07 FF 0D 0F FF FE
.:4830 1F FF FF 3F FF FF 7F FF
.:4838 FF FF FF FF FF FF FF 00
.:4840 80 00 00 C0 00 00 E8 80
.:4848 00 FD C0 00 DF E0 00 BF
.:4850 F1 00 7F FB 80 FF FF C0
.:4858 FF FF E0 FF FF F0 FF FF
```

```
.:4860 F8 FF FF F8 FF FF FC FF
.:4868 FF FE FF FF FF FF FF FF
.:4870 7F FF FF BF FF FF DF FF
.:4878 FF EF FF FF F7 FF FF 00
.:4880 00 00 00 00 00 00 00 00
.:4888 00 00 00 00 00 00 01 00
.:4890 00 23 00 00 77 00 00 FF
.:4898 00 01 FB 00 03 FD 00 07
.:48A0 FE 00 4F FF 00 FF FF 41
.:48A8 FF FF E3 FF FF F7 FF FF
.:48B0 FF FF FF EF FF FF DF FF
.:48B8 FF BF FF FF 7F FF FF 00
.:48C0 10 00 00 38 00 00 7C 00
.:48C8 00 FE 00 00 FF 00 00 FF
.:48D0 80 00 FF C0 00 FF E0 00
.:48D8 FF F0 00 FF F8 00 FF F8
.:48E0 00 7F FC 00 BF FE 00 DF
.:48E8 FF 00 EF FF 80 EF FF C0
.:48F0 F7 FF E0 FB FF F8 FB FF
.:48F8 FE FD FF FF FE FF FF 00
.:4900 18 00 00 08 00 00 08 00
.:4908 00 08 00 00 0E 00 00 0E
.:4910 00 00 0E 00 00 FF F0 00
.:4918 F3 E0 00 75 C0 00 73 00
.:4920 00 3E 00 00 00 00 00 00
.:4928 00 00 00 00 00 00 00 00
.:4930 00 00 00 00 00 00 00 00
.:4938 00 00 00 00 00 00 00 00
.:4940 00 10 00 00 78 00 00 F8
.:4948 00 03 FC 00 07 FE 00 07
.:4950 FF 00 0F FF 00 0F FF 80
.:4958 1F FF C0 1F FF C0 3F FF
.:4960 E0 3F FF E0 3F FF E0 7F
.:4968 FF E0 7F FF E0 FF FF F0
.:4970 FF FF F8 FF FF F8 FF FF
.:4978 F8 FF FF FC FF FF FC 00
.:4980 04 00 00 0E 00 00 0F 00
.:4988 00 1F 80 00 3F 80 00 37
.:4990 C0 00 6F E0 00 EB 70 00
.:4998 F7 B0 00 EF F0 00 FF F0
.:49A0 00 00 00 00 00 00 00 00
.:49A8 00 00 00 00 00 00 00 00
.:49B0 00 00 00 00 00 00 00 00
.:49B8 00 00 00 00 00 00 00 00
.:49C0 1B 00 00 08 00 00 18 00
.:49C8 00 3C 00 00 C2 00 00 C2
.:49D0 00 00 3C 00 00 18 00 00
.:49D8 08 00 00 18 00 00 00 00
.:49E0 00 00 00 00 00 00 00 00
.:49E8 00 00 00 00 00 00 00 00
.:49F0 00 00 00 00 00 00 00 00
.:49F8 00 00 00 00 00 00 00 00
.:4A00 0C 00 00 08 00 00 18 00
.:4A08 00 3C 00 00 C2 00 00 C2
.:4A10 00 00 3C 00 00 18 00 00
.:4A18 A8 00 00 58 00 00 00 00
.:4A20 00 00 00 00 00 00 00 00
.:4A28 00 00 00 00 00 00 00 00
.:4A30 00 00 00 00 00 00 00 00
```

```
.:4A38 00 00 00 00 00 00 00 00
.:4A40 59 00 00 49 00 00 2A 00
.:4A48 00 3E 00 00 2A 00 00 49
.:4A50 00 00 49 00 00 14 00 00
.:4A58 00 00 00 00 00 00 00 00
.:4A60 00 00 00 00 00 00 00 00
.:4A68 00 00 00 00 00 00 00 00
.:4A70 00 00 00 00 00 00 00 00
.:4A78 00 00 00 00 00 00 00 00
.:4A80 0C 00 00 6B 00 00 2A 00
.:4A88 00 7F 00 00 49 00 00 2A
.:4A90 00 00 6B 00 00 14 00 00
.:4A98 00 00 00 00 00 00 00 00
.:4AA0 00 00 00 00 00 00 00 00
.:4AA8 00 00 00 00 00 00 00 00
.:4AB0 00 00 00 00 00 00 00 00
.:4AB8 00 00 00 00 00 00 00 00
.:4AC0 0C 00 00 2A 00 00 2A 00
.:4AC8 00 7F 00 00 2A 00 00 2A
.:4AD0 00 00 2A 00 00 14 00 00
.:4AD8 00 00 00 00 00 00 00 00
.:4AE0 00 00 00 00 00 00 00 00
.:4AE8 00 00 00 00 00 00 00 00
.:4AF0 00 00 00 00 00 00 00 00
.:4AF8 00 00 00 00 00 00 00 00
.:4B00 18 00 00 08 00 00 2A 00
.:4B08 00 3E 00 00 2A 00 08 00
.:4B10 00 08 00 00 14 00 00 00
.:4B18 00 00 00 00 00 00 00 00
.:4B20 00 00 00 00 00 00 00 00
.:4B28 00 00 00 00 00 00 00 00
.:4B30 00 00 00 00 00 00 00 00
.:4B38 00 00 00 00 00 00 00 00
.:4B40 0C 00 00 6B 00 00 2A 00
.:4B48 00 7F 00 00 49 00 00 2A
.:4B50 00 00 6B 00 00 14 00 00
.:4B58 08 00 00 3C 00 00 00 00
.:4B60 00 00 00 00 00 00 00 00
.:4B68 00 00 00 00 00 00 00 00
.:4B70 00 00 00 00 00 00 00 00
.:4B78 00 00 00 00 00 00 00 00
.:4B80 0C 00 00 6B 00 00 2A 00
.:4B88 00 7F 00 00 49 00 00 2A
.:4B90 00 00 6B 00 00 14 00 00
.:4B98 08 00 00 3C 00 00 00 00
.:4BA0 00 00 00 00 00 00 00 00
.:4BA8 00 00 00 00 00 00 00 00
.:4BB0 00 00 00 00 00 00 00 00
.:4BB8 00 00 00 00 00 00 00 00
.:4BC0 01 00 00 03 80 00 1F C0
.:4BC8 00 3F E0 00 FF F0 00 FF
.:4BD0 E0 00 7F E0 00 7F C0 00
.:4BD8 3F 80 00 07 80 00 03 00
.:4BE0 00 00 00 00 00 00 00 00
.:4BE8 00 00 00 00 00 00 00 00
.:4BF0 00 00 00 00 00 00 00 00
.:4BF8 00 00 00 00 00 00 00 00
.:4C00 FF FF 00 00 FF FF 02 00
```

Listing C-26: The BOGDEF Source Code

```
1000 ;PUT"@0:BOGDEF"
1010 ;GET"CAST"
1020 ;
1030 ;
1040 ;BOG HOP RAM DEFINITIONS
1050 ;BOGDEF
1060 ;
1070 ;LOCATIONS 0 & 1 ARE HARDWARE
1080 ;SO USABLE RAM STARTS AT $02
1090 ;
1100 *      = $02
1110 S1CORG DS 1              ;SID CHIP SHADOW
1120 S2CORG DS 1              ;REGISTERS
1130 S3CORG DS 1
1140 S1FRLO DS 1
1150 S1FRHI DS 1
1160 S2FRLO DS 1
1170 S2FRHI DS 1
1180 S3FRLO DS 1
1190 S3FRHI DS 1
1200 S1PWLO DS 1
1210 S1PWHI DS 1
1220 S2PWLO DS 1
1230 S2PWHI DS 1
1240 S3PWLO DS 1
1250 S3PWHI DS 1
```

```
1260 S1ATDC DS 1
1270 S2ATDC DS 1
1280 S3ATDC DS 1
1290 S1SURL DS 1
1300 S2SURL DS 1
1310 S3SURL DS 1
1320 FILLO  DS 1
1330 FILHI  DS 1
1340 MMOD   DS 1
1350 RFIL   DS 1                 ;END OF SHADOWS
1360 ;
1370 SNDTM1 DS 1                 ;SOFTWARE TIMERS
1380 SNDTM2 DS 1
1390 SNDTM3 DS 1
1400 ;
1410 SRC    DS 2                 ;POINTERS FOR DATA
1420 DST    DS 2                 ;MOVEMENT
1430 SCRPT  DS 2
1440 SCRDST DS 2
1450 ;
1460 NOTPT1 DS 2                 ;POINTERS TO NOTES
1470 NOTPT2 DS 2
1480 NOTPT3 DS 2
1490 NOTTM1 DS 2                 ;POINTERS TO NOTE TIMES
1500 NOTTM2 DS 2
1510 NOTTM3 DS 2
1520 OPTION DS 1
1530 RAND1  DS 1                 ;RANDOM NUMBER REGISTERS
1540 RAND2  DS 1
1550 RAND3  DS 1
1560 RAND4  DS 1
1570 LEVEL  DS 1                 ;CURRENT LEVEL OF PLAY
1580 ;
1590 SCREEN DS 1                 ;SYSTEM TIMING REGISTERS
1600 RANSEC DS 1
1610 SECOND DS 1
1620 ENABLE DS 1
1630 ;
1640 BUF    DS 8                 ;TEMPORARY DATA BUFFERS
1650 BUF1   DS 8
1660 IBUF   DS 8
1670 MBUF   DS 8
1680 LPCNT1 DS 1                 ;LOOP COUNTERS
1690 LPCNT2 DS 1
1700 LPCNT3 DS 1
1710 LPCNT4 DS 1
1720 ;
1730 ;
1740 COLOR  DS 1                 ;THESE REGISTERS ARE
1750 POINT  DS 2                 ;USED BY THE PLOTTING
1760 CTEMP  DS 1                 ;ROUTINE
1770 GBASE  DS 2
1780 YPNT   DS 1                 ;Y-COORDINATE
1790 XPNT   DS 1                 ;X-COORDINATE
1800 ;
1810 HMSB   DS 1                 ;9'TH BIT REGISTER
1820 ;
1830 ;
1840 HORNC  DS 1                 ;HORIZONTAL INCREMENT
```

```
1850 VERNC  DS 1          ;VERTICAL INCREMENT
1860 ;
1870 LIVES  DS 1          ;NUMBER OF LIVES LEFT
1880 SCORE  DS 4          ;SCORE COUNTER
1890 MOUNTV DS 8          ;MOUNTAIN VERTICAL
1900 MOUNTH DS 8          ;MOUNTAIN HORIZONTAL
1910 MOUNTP DS 8          ;SPRITE POINTERS
1920 MOUNTC DS 8          ;MOUNTAIN COLORS
1930 ;
1940 MNTMSB DS 1          ;EXTRA MSB--MOUNTAIN
1950 ;
1960 ;
1970 WHOLIV DS 1          ;WHICH BAD GUY LEFT
1980 TMSCOL DS 1          ;TEMP SSCOL
1990 TMBCOL DS 1          ;TEMP SBCOL
2000 ;
2010 ;
2020 ;
2030 PLAYH  DS 1          ;PLAYER HORIZONTAL
2040 MEANH  DS 7          ;BAD GUY HORIZONTAL
2050 ;
2060 PLAYV  DS 1          ;PLAYER VERTICAL
2070 MEANV  DS 7          ;BAD GUY VERTICAL
2080 ;
2090 PLAYC  DS 1          ;PLAYER COLOR
2100 MEANC  DS 7          ;BAD GUY COLOR
2110 ;
2120 PLAYD  DS 1          ;PLAYER DIRECTION
2130 MEAND  DS 7          ;BAD GUY DIRECTION
2140 ;
2150 PLAYE  DS 1          ;PLAYER EXPLOSION
2160 MEANE  DS 7          ;BAD GUY EXPLOSION
2170 ;
2180 PLAYP  DS 1          ;PLAYER SPRITE POINT
2190 MEANP  DS 7          ;BAD GUY POINTER
2200 ;
2210 MEANMV DS $10        ;MOVEMENT CHART
2220 ;
2230 ;
2240 SHOTS  DS 1          ;SHOTS IN FLIGHT
2250 ;
2260 SHOTSH DS 8          ;SHOT HORIZONTAL
2270 SHOTSV DS 8          ;SHOT VERTICAL
2280 SHOTSD DS 8          ;SHOT DIRECTION
2290 ;
2300 PLAYSP DS 1          ;PLAYER SPEED
2310 BADSPH DS 1          ;BAD GUY SPEED-HORIZ
2320 BADSPV DS 1          ;BAD GUY SPEED-VERT
2330 ;
2340 ;
2350 ;
2360        .END
```

Listing C-27: The BOGDAT Source Code

```
1000 ;PUT"@0:BOGDAT"
1010 ;LOAD"ASM",8
1020 ;
1030 ;THIS IS BOGDAT--THE DATA FILE
```

```
1040 ;FOR BOG HOP
1050 ;
1060 BITPOS .BYTE $01,$02,$04,$08,$10,$20,$40,$80
1070 BITAND .BYTE $FE,$FD,$FB,$F7,$EF,$DF,$BF,$7F
1080 ANDPOS .BYTE $FE,$FD,$FB,$F7,$EF,$DF,$BF,$7F
1090 ;
1100 TYPELK .BYTE $00,$04,$08,$0C,$10,$14
1110 ;
1120 ;
1130 ;
1140 ;
1150 ;
1160 ;MOVEMENT PATTERNS FOR DIFFERENT
1170 ;TYPES OF CHARACTERS
1180 ;
1190 BEE1    .BYTE 4,4,6,2,$A,9,5,4,4,6,6,4,5,5,1,9,8,$A,2,6,6,6,4,4
1200         .BYTE 5,5,1,5,9,8,9,8,$A,$A,2,6,6,4,6,2
1210         .BYTE 4,4,6,2,$A,9,5,4,4,6,6,4,5,5,1,9,8,$A,2,6,6,6,4,4
1220         .BYTE 5,5,1,5,9,8,9,8,$A,$A,2,6,6,4,6,2
1230         .BYTE $00,$00,$00
1240 ;
1250 BEE2    .BYTE 4,6,6,6,6,6,6,6,2,6,2,6,2,$A,2,$A,$A,$A,8,$A,8,$A
1260         .BYTE 8,9,8,9,9,1,9,1,5,1,5,5,1,1,5,4,6,6,6
1270         .BYTE $00,$00,$00
1280 ;
1290 BEE3    .BYTE 6,6,6,4,4,4,5,5,5,1,5,5,1,1,9,9,9,8,9,8,8,$A,8
1300         .BYTE $A,8,$A,$A,2,$A,2,2,6,6,2,6,6,4,6,4
1310         .BYTE $00,$00,$00
1320 ;
1330 BEE4    .BYTE 5,5,8,8,$A,$A,2,2,6,6,4,4,4,6,6,2,$A,$A,$A,8,8,9
1340         .BYTE 1,9,1,5,5,5,1,5,4,5,5,1,9,8,$A,2,6,2
1350         .BYTE 5,5,8,8,$A,$A,2,2,6,6,4,4,4,6,6,2,$A,$A,$A,8,8,9
1360         .BYTE 1,9,1,5,5,5,1,5,4,5,5,1,9,8,$A,2,6,2
1370         .BYTE $00,$00,$00
1380 ;
1390 ;
1400 ;
1410 BIRD1   .BYTE 4,6,4,4,6,4,4,6,4,4,5,4,4,5,5,4,5,5,5,1,5,5,1,5
1420         .BYTE 5,4,4,6,6,2,6,2,6,6,4,6,6,4,6,4
1430         .BYTE 5,4,4,6,6,2,6,2,6,6,4,6,6,4,6,4
1440         .BYTE $00,$00,$00
1450 ;
1460 BIRD2   .BYTE 4,6,4,6,4,6,4,6,4,6,4,6,6,2,6,2,6,2,6,2,5,1,5,1
1470         .BYTE 5,1,5,1,5,4,5,4,5,4,5,4,5,4,5,4
1480         .BYTE 5,1,5,1,5,4,5,4,5,4,5,4
1490         .BYTE $00,$00,$00
1500 ;
1510 BIRD3   .BYTE 4,5,4,5,5,5,5,1,5,1,5,1,5,1,5,1,5,1,5,1,4,6,6,2
1520         .BYTE 6,2,2,6,2,2,6,4,5,5,4,6,6,4,5,5
1530         .BYTE $00,$00,$00
1540 ;
1550 BIRD4   .BYTE 4,5,5,1,1,5,1,5,4,6,2,6,2,6,6,4,4,5,5,1,1,6,6,4
1560         .BYTE 6,6,2,2,6,4,5,5,5,4,4,6,6,6,2,6
1570         .BYTE $00,$00,$00
1580 ;
1590 ;
1600 ;
1610 BAT1    .BYTE 4,5,4,6,4,5,4,5,1,5,1,4,1,5,4,5,2,5,4,6,4,6,2,6
1620         .BYTE 4,6,2,6,2,6,4,6,4,6,4,5,1,5,4,6
```

```
1630          .BYTE $00,$00,$00
1640 ;
1650 BAT2     .BYTE 1,5,1,1,2,1,1,4,1,5,8,1,5,4,6,4,6,2,8,2,4,2,1,6
1660          .BYTE 2,6,4,2,5,1,4,8,1,5,4,6,6,2,6,4
1670          .BYTE $00,$00,$00
1680 ;
1690 BAT3     .BYTE 2,5,2,6,2,$A,9,$A,2,1,2,$A,4,$A,2,8,1,8,2,9,8,9
1700          .BYTE 4,1,9,1,5,9,1,4,5,1,4,6,5,4,1,2,4,6
1710          .BYTE $00,$00,$00
1720 ;
1730 BAT4     .BYTE 4,5,4,1,5,2,5,1,9,5,1,9,8,$A,9,5,$A,8,$A,2,$A,8,2,6
1740          .BYTE 4,1,5,9,1,5,4,6,4,6,2,8,2,6,4,5
1750          .BYTE $00,$00,$00
1760 ;
1770 ;
1780 ;
1790 FROG1    .BYTE 6,4,6,4,3,3,3,6,2,6,2,6,2,6,3
1800          .BYTE 3,3,3,5,5,5,5,5,3,3,3,3,4,4,4,3,3,3,5,1,1,5,3,3,3,3
1810          .BYTE $00,$00,$00
1820 ;
1830 FROG2    .BYTE 4,6,4,4,6,4,4,6,3,3,3,3,5,5,5,5,5,3,3,3,3
1840          .BYTE 5,1,1,5,1,1,5,1,1,3,3,3,4,4,4,3,3,3,3,3
1850          .BYTE $00,$00,$00
1860 ;
1870 FROG3    .BYTE 1,1,1,3,3,3,3,3,$A,$A,$A,$A,$A,$A,$A
1880          .BYTE 3,3,3,8,9,8,9,8,9,8,9,3,3,3,$A,2,2,$A,2,2,$A,3,3,3,3
1890          .BYTE $00,$00,$00
1900 ;
1910 FROG4    .BYTE 8,8,8,8,3,3,3,9,9,9,3,3,3,$A,$A,2,$A,$A,3,3,3,3
1920          .BYTE 2,2,2,2,3,3,3,5,5,5,3,3,3,3,3
1930          .BYTE $00,$00,$00
1940 ;
1950 ;
1960 ;
1970 SNAKE1   .BYTE 6,6,6,6,6,5,1,5,5,1,5,5,1,5,6,6,5,5,5,5,2,6,2,6
1980          .BYTE 2,6,2,5,5,2,6,6,1,5,1,5,1,5,6,6
1990          .BYTE $00,$00,$00
2000 ;
2010 SNAKE2 .BYTE 6,2,6,2,5,5,5,6,6,6,6,6,5,1,5,1,5,1,6,6,5,5
2020          .BYTE 1,5,6,2,6,6,2,6,6,2,6,6,2,5,5,5
2030          .BYTE $00,$00,$00
2040 ;
2050 ;
2060 ;
2070 ;
2080 ;DIFFERENT CHARACTERS HAVE
2090 ;DIFFERENT MOVEMENT PATTERNS.
2100 ;THE FOLLOWING ADDRESS CHARTS
2110 ;ALLOW THE PROPER MOVEMENT TO BE
2120 ;SELECTED FOR THE CHARACTER.
2130 ;
2140 BEELK    .WORD BEE1,BEE2,BEE3,BEE4
2150          .WORD $0000,$0000
2160 ;
2170 BIRDLK .WORD BIRD1,BIRD2,BIRD3,BIRD4
2180          .WORD $0000,$0000
2190 ;
2200 BATLK    .WORD BAT1,BAT2,BAT3,BAT4
2210          .WORD $0000,$0000
```

```
2220 ;
2230 FROGLK .WORD FROG1,FROG2,FROG3,FROG4
2240        .WORD $0000,$0000
2250 ;
2260 SNAKLK .WORD SNAKE1,SNAKE2,SNAKE2,SNAKE1
2270        .WORD $0000,$0000
2280 BASLK  .WORD BEELK,BIRDLK,BATLK,FROGLK,SNAKLK
2290 ;
2300 ;
2310 ;
2320 ;BY CHANGING THE PATTERN LOOKUP
2330 ;TABLES, THE CHARACTER WILL
2340 ;MOVE DIFFERENTLY.
2350 ;
2360 ;
2370 ;BADSEQ IS THE SEQUENCE IN WHICH
2380 ;THE DIFFERENT CHARACTERS WILL
2390 ;APPEAR ON THE SCREEN.
2400 ;
2410 ;
2420 BADSEQ .BYTE 0,1,2,3,4,1,0,4,3,2,1,3,2,0,4,3,2,1,0,0
2430 ;
2440 ;
2450 ;THE FOLLOWING LOOKUP CHART IS
2460 ;USED TO FIND THE BASE ADDRESS
2470 ;OF THE CHARACTER SET FOR THE
2480 ;NUMBERS. THIS SET IS FOR USE IN
2490 ;A HIGH RESOLUTION MULTI-COLOR
2500 ;MODE.
2510 ;
2520 ;THE NUMBER SET IS IN THE FILE--
2530 ;"LKUP"--WHICH MUST BE LOADED
2540 ;PRIOR TO RUNNING THE PROGRAM.
2550 ;
2560 NUMBER .WORD ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX
2570        .WORD SEVEN,EIGHT,NINE,NUL,$0000
2580 ;
2590 ;
2600 ;
2610 ;THE FOLLOWING TWO CHARTS SET THE
2620 ;VERTICAL AND HORIZONTAL SPEEDS
2630 ;OF THE CHARACTERS DEPENDING
2640 ;ON THE LEVEL OF PLAY
2650 ;
2660 SPEEDH .BYTE 1,1,1,1,1,1,2,2,2,3,3,4,3,4,3,5,6
2670 ;
2680 SPEEDV .BYTE 1,2,1,1,1,1,2,3,2,2,3,5,2,5,4,6,7
2690 ;
2700 ;
2710 ;THE FOLLOWING CHART IS USED TO
2720 ;DETERMINE HOW OFTEN THE BAD GUYS
2730 ;WILL FIRE. A VALUE OF ZERO WILL
2740 ;NOT LET ANY BAD GUYS SHOOT
2750 ;
2760 FIRPOW .BYTE 1,2,4,4,5,8,8,6,6,5,8,6,9,6,6,8,9
2770 ;
2780 ;
2790 SHTDR  .BYTE $04,$06,$05,$04
2800 ;
```

```
2810 COLK    .BYTE $00,$55,$AA,$FF
2820 POR     .BYTE $C0,$30,$0C,$03
2830 ;
2840         .END
```

Listing C-28: The XXPLOT Subroutine Source Code

```
1000 ;PUT"@0:XXPLOT
1010 ;Y IN YPNT
1020 ;X/2 IN XPNT
1030 ;COLOR IN COLOR
1040 ;GRAPHICS BASE ADDRESS IN GBASE
1050 ;
1060 ;EXCLUSIVE OR'S A POINT
1070 ;
1080 ;SINCE THE INPUT TO THIS ROUTINE
1090 ;WILL BE SPRITE POSITIONS, THE
1100 ;XPNT AND YPNT REGISTERS WILL
1110 ;BE ADJUSTED FOR THE SPRITE OFFSET
1120 ;
1130 XPLOT  ANOP
1140        LDA YPNT
1150        SEC
1160        SBC #$32         ;VERTICAL OFFSET
1170        CMP #$C7
1180        BCS RTR1
1190        TAX
1200        LDA XPNT
1210        SEC
1220        SBC #$0E         ;HORIZONTAL OFFSET/2
1230        CMP #$A0
1240        BCS RTR1
1250        LSR A
1260        LSR A
1270        TAY
1280        LDA GBASE
1290        CLC
1300        ADC VLKUPL,X
1310        STA POINT
1320        LDA GBASE+1
1330        ADC VLKUPH,X
1340        STA POINT+1
1350        LDA POINT
1360        CLC
1370        ADC HLKUPL,Y
1380        STA POINT
1390        LDA POINT+1
1400        ADC HLKUPH,Y
1410        STA POINT+1
1420 ;
1430        LDA XPNT
1440        AND #$03         ;STRIP BIT POS
1450        TAX
1460        LDY #$00
1470        LDA (POINT),Y
1480 ;AND PAND,X ;CLEAR BIT
```

```
1490          STA CTEMP
1500          LDY COLOR
1510          LDA COLK,Y
1520          AND POR,X
1530          EOR CTEMP
1540          LDY #$00
1550          STA (POINT),Y
1560 RTR1     RTS
1570 ;
1580 ;
1590          .END
```

Listing C-29: The LOOKUP Data File

```
.:1000 28 43 29 20 43 4F 50 59
.:1008 52 49 47 48 54 20 31 39
.:1010 38 34 2C 20 53 54 45 56
.:1018 45 4E 20 42 52 45 53 53
.:1020 28 50 29 20 50 45 52 46
.:1028 4F 52 4D 41 4E 43 45 20
.:1030 31 39 38 34 2C 20 53 54
.:1038 45 56 45 4E 20 42 52 45
.:1040 53 53 3C C0 C3 CC F0 C0
.:1048 3F 00 00 C0 C0 C0 C0 C0
.:1050 00 00 0C 3C 0C 0C 0C 0C
.:1058 3F 00 00 00 00 00 00 00
.:1060 00 00 3F C0 00 0F 30 C0
.:1068 FF 00 00 C0 C0 00 00 00
.:1070 C0 00 FF 00 03 0F 00 C0
.:1078 3F 00 C0 C0 00 00 C0 C0
.:1080 00 00 03 0F 33 C3 FF 03
.:1088 03 00 00 00 00 00 C0 00
.:1090 00 00 FF C0 FF 00 00 C0
.:1098 3F 00 C0 00 00 C0 C0 C0
.:10A0 00 00 0F 30 C0 FF C0 C0
.:10A8 3F 00 C0 00 00 00 C0 C0
.:10B0 00 00 FF 00 03 0C 30 30
.:10B8 30 00 C0 C0 00 00 00 00
.:10C0 00 00 3F C0 C0 3F C0 C0
.:10C8 3F 00 00 C0 C0 00 C0 C0
.:10D0 00 00 3F C0 C0 3F 00 03
.:10D8 FC 00 00 C0 C0 C0 C0 00
.:10E0 00 00 00 00 00 00 00 00
.:10E8 00 00 00 00 00 00 00 00
.:10F0 00 00 00 28 50 78 A0 C8
.:10F8 F0 18 40 68 90 B8 E0 08
.:1100 30 58 80 A8 D0 F8 20 48
.:1108 70 98 C0 04 04 04 04 04
.:1110 04 04 05 05 05 05 05 05
.:1118 06 06 06 06 06 06 06 06
.:1120 07 07 07 07 07 00 01 02
.:1128 03 04 05 06 07 40 41 42
.:1130 43 44 45 46 47 80 81 82
.:1138 83 84 85 86 87 C0 C1 C2
.:1140 C3 C4 C5 C6 C7 00 01 02
.:1148 03 04 05 06 07 40 41 42
.:1150 43 44 45 46 47 80 81 82
.:1158 83 84 85 86 87 C0 C1 C2
.:1160 C3 C4 C5 C6 C7 00 01 02
.:1168 03 04 05 06 07 40 41 42
.:1170 43 44 45 46 47 80 81 82
.:1178 83 84 85 86 87 C0 C1 C2
.:1180 C3 C4 C5 C6 C7 00 01 02
.:1188 03 04 05 06 07 40 41 42
.:1190 43 44 45 46 47 80 81 82
.:1198 83 84 85 86 87 C0 C1 C2
.:11A0 C3 C4 C5 C6 C7 00 01 02
.:11A8 03 04 05 06 07 40 41 42
.:11B0 43 44 45 46 47 80 81 82
.:11B8 83 84 85 86 87 C0 C1 C2
.:11C0 C3 C4 C5 C6 C7 00 01 02
.:11C8 03 04 05 06 07 40 41 42
.:11D0 43 44 45 46 47 80 81 82
.:11D8 83 84 85 86 87 C0 C1 C2
.:11E0 C3 C4 C5 C6 C7 00 01 02
.:11E8 03 04 05 06 07 40 41 42
.:11F0 43 44 45 46 47 80 81 82
.:11F8 83 84 85 86 87 00 00 00
.:1200 00 00 00 00 00 01 01 01
.:1208 01 01 01 01 01 02 02 02
.:1210 02 02 02 02 02 03 03 03
.:1218 03 03 03 03 03 05 05 05
.:1220 05 05 05 05 05 06 06 06
.:1228 06 06 06 06 06 07 07 07
.:1230 07 07 07 07 07 08 08 08
.:1238 08 08 08 08 08 0A 0A 0A
.:1240 0A 0A 0A 0A 0A 0B 0B 0B
.:1248 0B 0B 0B 0B 0B 0C 0C 0C
.:1250 0C 0C 0C 0C 0C 0D 0D 0D
.:1258 0D 0D 0D 0D 0D 0F 0F 0F
.:1260 0F 0F 0F 0F 0F 10 10 10
.:1268 10 10 10 10 10 11 11 11
.:1270 11 11 11 11 11 12 12 12
.:1278 12 12 12 12 12 14 14 14
.:1280 14 14 14 14 14 15 15 15
.:1288 15 15 15 15 15 16 16 16
.:1290 16 16 16 16 16 17 17 17
.:1298 17 17 17 17 17 19 19 19
.:12A0 19 19 19 19 19 1A 1A 1A
.:12A8 1A 1A 1A 1A 1A 1B 1B 1B
.:12B0 1B 1B 1B 1B 1B 1C 1C 1C
.:12B8 1C 1C 1C 1C 1C 1E 1E 1E
.:12C0 1E 1E 1E 1E 1E 1F 1F 1F
.:12C8 1F 1F 1F 1F 1F 00 08 10
```

```
.:12D0 18 20 28 30 38 40 48 50
.:12D8 58 60 68 70 78 80 88 90
.:12E0 98 A0 A8 B0 B8 C0 C8 D0
.:12E8 D8 E0 E8 F0 F8 00 08 10
.:12F0 18 20 28 30 38 00 00 00
.:12F8 00 00 00 00 00 00 00 00
.:1300 00 00 00 00 00 00 00 00
.:1308 00 00 00 00 00 00 00 00
.:1310 00 00 00 00 00 01 01 01
.:1318 01 01 01 01 01 FF 00 72
.:1320 00 F1 00 FF 00 FF 00 FF
.:1328 00 FF 00 FF 00 FF 2F FF
.:1330 00 70 00 FF 00 FF 00 FF
.:1338 00 FF 00 FF 00 FF 4F F7
.:1340 00 72 00 DD 00 FF 00 58
.:1348 00 FF 00 EF 00 7D 00 FF
.:1350 00 FF 00 FF 00 FF 00 DF
.:1358 00 FF 00 FF 00 FF 00 F0
.:1360 00 7F 00 7D 20 FF 52 FF
```

```
.:1368 00 FF 00 FF 00 FF 00 FF
.:1370 00 FF 00 FF 00 FF 00 FF
.:1378 00 FF 00 FF 00 FF E0 F7
.:1380 00 DF 00 85 02 FF 00 DD
.:1388 00 FF 00 FF D0 FF 02 DF
.:1390 00 FF 00 FF 00 FF 00 FF
.:1398 00 FF 00 FF 00 FF 00 FF
.:13A0 00 FF 00 FF 7D DD 02 FF
.:13A8 00 FF 00 FF 00 FF D1 FF
.:13B0 00 7F 00 FF 00 D5 22 FF
.:13B8 00 FF 00 FF 00 FF 00 FF
.:13C0 05 FF 00 FF 00 FF 00 FD
.:13C8 00 FF 00 FF 00 FF 00 FF
.:13D0 00 FF 00 FF 00 FF 00 FF
.:13D8 00 FF 00 2F C0 FF 00 FF
.:13E0 00 FF 00 FF 00 FF 00 FF
.:13E8 DD FF 00 FF 00 FF 00 F6
.:13F0 4C C8 15 EE 5D 08 EE 5D
.:13F8 08 D0 39 AD 51 08 C9 FF
```

Listing C-30: The BOGSPR File

```
B*
   PC  SR AC XR YR SP
.;C03E 32 00 C3 00 F6
.


.:4000 20 00 00 8C 00 00 82 00
.:4008 20 80 FA 80 3A AE 00 00
.:4010 EA C0 0A 08 20 0E 08 0C
.:4018 00 02 A0 00 00 00 00 00
.:4020 00 00 00 00 00 00 00 00
.:4028 00 00 00 00 00 00 00 00
.:4030 00 00 00 00 00 00 00 00
.:4038 00 00 00 00 00 00 00 00
.:4040 2C 00 00 82 00 00 82 00
.:4048 20 20 80 80 20 8E 00 0A
.:4050 AB A8 00 EA 02 0E B8 02
.:4058 0A 02 BC 00 00 00 00 00
.:4060 00 00 00 00 00 00 00 00
.:4068 00 00 00 00 00 00 00 00
.:4070 00 00 00 00 00 00 00 00
.:4078 00 00 00 00 00 00 00 00
.:4080 20 00 00 8F 00 20 80 80
.:4088 80 CA 8F 00 20 FA 00 00
.:4090 AA C0 0A A0 08 0B 08 02
.:4098 00 03 C2 00 00 28 00 00
.:40A0 00 00 00 00 00 00 00 00
.:40A8 00 00 00 00 00 00 00 00
.:40B0 00 00 00 00 00 00 00 00
.:40B8 00 00 00 00 00 00 00 00
.:40C0 00 00 20 0A 8E 80 30 AB
.:40C8 00 80 EB C0 C0 B8 20 2B
.:40D0 0C 08 0A 02 08 00 00 F0
.:40D8 00 00 00 00 00 00 00 00
.:40E0 00 00 00 00 00 00 00 00
.:40E8 00 00 00 00 00 00 00 00
```

```
.:40F0 00 00 00 00 00 00 00 00
.:40F8 00 00 00 00 00 00 00 00
.:4100 01 00 80 0A 02 60 26 49
.:4108 A4 60 AA 0A 90 18 09 00
.:4110 20 00 02 48 00 02 08 00
.:4118 00 00 00 00 00 00 00 00
.:4120 00 00 00 00 00 00 00 00
.:4128 00 00 00 00 00 00 00 00
.:4130 00 00 00 00 00 00 00 00
.:4138 00 00 00 00 00 00 00 00
.:4140 09 00 60 1A 82 98 A0 41
.:4148 04 40 8A 0A 00 24 01 00
.:4150 28 02 00 60 00 02 A0 00
.:4158 01 20 00 00 18 00 00 80
.:4160 00 00 00 00 00 00 00 00
.:4168 00 00 00 00 00 00 00 00
.:4170 00 00 00 00 00 00 00 00
.:4178 00 00 00 00 00 00 00 00
.:4180 80 00 06 10 00 28 24 00
.:4188 90 02 01 A0 06 0A 40 01
.:4190 26 00 02 68 00 00 90 00
.:4198 00 80 00 01 00 00 0A 40
.:41A0 00 20 80 00 02 20 00 01
.:41A8 08 00 00 00 00 00 00 00
.:41B0 00 00 00 00 00 00 00 00
.:41B8 00 00 00 00 00 00 00 00
.:41C0 00 00 60 00 02 A4 06 01
.:41C8 18 29 8A 08 18 98 02 A0
.:41D0 64 01 40 20 00 00 10 00
.:41D8 00 60 00 06 A0 00 08 84
.:41E0 00 00 48 00 00 00 00 00
.:41E8 00 00 00 00 00 00 00 00
.:41F0 00 00 00 00 00 00 00 00
.:41F8 00 00 00 00 00 00 00 00
.:4200 0A 82 A0 A0 28 0A 00 A0
.:4208 00 00 80 00 00 00 00 00
```

```
.:4210 00 00 00 00 00 00 00 00
.:4218 00 00 00 00 00 00 00 00
.:4220 00 00 00 00 00 00 00 00
.:4228 00 00 00 00 00 00 00 00
.:4230 00 00 00 00 00 00 00 00
.:4238 00 00 00 00 00 00 00 00
.:4240 0A 00 A0 A0 82 08 00 28
.:4248 02 00 A0 00 00 20 00 00
.:4250 00 00 00 00 00 00 00 00
.:4258 00 00 00 00 00 00 00 00
.:4260 00 00 00 00 00 00 00 00
.:4268 00 00 00 00 00 00 00 00
.:4270 00 00 00 00 00 00 00 00
.:4278 00 00 00 00 00 00 00 00
.:4280 A0 00 0A 08 00 A0 02 0A
.:4288 00 00 A0 00 02 80 00 00
.:4290 80 00 00 00 00 00 00 00
.:4298 00 00 00 00 00 00 00 00
.:42A0 00 00 00 00 00 00 00 00
.:42A8 00 00 00 00 00 00 00 00
.:42B0 00 00 00 00 00 00 00 00
.:42B8 00 00 00 00 00 00 00 00
.:42C0 00 00 A0 0A 02 08 20 A8
.:42C8 02 80 20 00 00 A0 00 00
.:42D0 00 00 00 00 00 00 00 00
.:42D8 00 00 00 00 00 00 00 00
.:42E0 00 00 00 00 00 00 00 00
.:42E8 00 00 00 00 00 00 00 00
.:42F0 00 00 00 00 00 00 00 00
.:42F8 00 00 00 00 00 00 00 00
.:4300 08 00 00 0C 00 00 2E 28
.:4308 00 2E A8 00 2A AB 00 AA
.:4310 AB 00 AA AB C0 AA AA 80
.:4318 2A AA 80 02 AA 80 03 EA
.:4320 80 02 EA 80 02 EF 80 02
.:4328 0F 00 02 0B 00 02 0B C0
.:4330 02 02 80 02 0A 80 0A 28
.:4338 00 00 00 00 00 00 00 00
.:4340 08 00 00 08 00 00 2A 00
.:4348 00 2E 00 00 2A 80 00 AE
.:4350 A0 00 AA AC 00 AA AC 00
.:4358 2A AF 00 03 AA 00 03 EA
.:4360 00 02 EA 00 0A EA 00 08
.:4368 3E 00 20 0C 00 20 0C 00
.:4370 80 0E 00 00 02 00 00 02
.:4378 00 00 08 00 00 08 00 00
.:4380 02 00 00 02 0B 00 0B 7B
.:4388 00 2B AB C0 2A AA 80 2A
.:4390 AA 80 0A EA 80 02 FA C0
.:4398 00 BA C0 0A B8 F0 20 00
.:43A0 30 80 00 08 00 00 02 00
.:43A8 00 00 00 00 00 00 00 00
.:43B0 00 00 00 00 00 00 00 00
.:43B8 00 00 00 00 00 00 00 00
.:43C0 08 00 00 0A 2C 00 2A AC
.:43C8 00 2E AF 00 AA AA 00 AA
.:43D0 AA 80 AB AA 80 2B EA 80
.:43D8 02 EA 80 02 EB 00 02 03
.:43E0 00 02 03 00 08 03 80 08
.:43E8 00 80 08 00 80 20 02 00
.:43F0 00 00 00 00 00 00 00 00
.:43F8 00 00 00 00 00 00 00 00
.:4400 28 00 00 2A 00 00 42 00
.:4408 00 03 00 00 02 00 00 02
.:4410 00 00 01 00 00 0A 00 00
.:4418 09 20 00 08 20 00 18 AC
.:4420 00 20 80 40 30 80 80 23
.:4428 88 00 22 02 00 31 02 00
.:4430 23 2C 00 22 20 20 21 20
.:4438 C0 2E 20 80 08 0A 00 00
.:4440 28 00 00 AD 00 00 82 00
.:4448 00 C1 00 00 02 00 00 02
.:4450 00 00 03 00 00 02 00 00
.:4458 02 02 00 02 03 80 01 08
.:4460 80 0A 0C 80 08 08 70 08
.:4468 28 80 0C 28 80 08 30 20
.:4470 08 20 10 08 20 20 04 10
.:4478 20 0A 80 12 02 80 04 00
.:4480 08 00 00 3E 00 00 22 00
.:4488 00 B1 80 00 80 80 00 C0
.:4490 80 00 00 40 00 00 80 00
.:4498 00 80 00 00 C0 00 00 80
.:44A0 00 00 70 20 00 20 88 00
.:44A8 10 8C 00 20 C8 00 20 81
.:44B0 00 22 02 00 32 02 00 23
.:44B8 02 00 2A 03 90 04 00 00
.:44C0 38 00 00 A8 00 00 24 00
.:44C8 00 08 00 00 08 00 00 0C
.:44D0 00 00 08 00 00 08 00 00
.:44D8 02 00 00 02 02 00 03 02
.:44E0 00 02 04 80 01 08 80 02
.:44E8 08 C0 02 0C 80 02 08 80
.:44F0 03 08 41 02 28 22 02 10
.:44F8 22 02 A0 23 00 80 04 00
.:4500 00 50 00 00 70 00 00 70
.:4508 00 00 C0 00 00 C0 00 02
.:4510 80 00 0E 80 00 32 B0 00
.:4518 32 8C 00 32 83 C0 32 80
.:4520 00 02 80 00 02 80 00 02
.:4528 20 00 08 08 00 08 08 00
.:4530 20 08 00 20 08 00 80 08
.:4538 00 20 0A 00 00 00 00 00
.:4540 00 50 00 00 50 00 00 70
.:4548 00 00 70 00 00 C0 00 02
.:4550 80 00 03 B0 00 0E B0 00
.:4558 32 8C 00 32 8C 00 32 83
.:4560 00 32 80 00 02 80 00 02
.:4568 20 00 02 20 00 02 08 00
.:4570 08 08 00 A0 08 00 80 08
.:4578 00 0A 00 00 00 00 00 00
.:4580 00 50 00 00 70 00 00 70
.:4588 00 00 C0 00 00 C0 00 00
.:4590 80 00 02 80 00 03 80 00
.:4598 0E 80 00 0E 80 00 0F 80
.:45A0 00 02 BC 00 02 80 00 02
.:45A8 80 00 02 80 00 02 20 00
.:45B0 08 20 00 20 20 00 20 20
.:45B8 00 00 28 00 00 00 00 00
```

```
.:45C0 00 50 00 00 70 00 00 70
.:45C8 00 00 C0 00 00 C0 00 00
.:45D0 80 00 03 80 00 0D B0 00
.:45D8 32 8C 00 32 83 C0 32 80
.:45E0 00 32 80 00 02 80 00 02
.:45E8 80 00 00 80 00 00 A0 00
.:45F0 02 20 00 02 08 00 08 08
.:45F8 80 08 02 00 02 00 00 00
.:4600 00 00 00 00 00 00 00 00
.:4608 00 00 00 00 03 00 00 03
.:4610 F0 00 0F 70 00 0D 7C 00
.:4618 03 5C 00 03 FC 00 00 C0
.:4620 00 00 00 00 00 00 00 00
.:4628 00 00 00 00 00 00 00 00
.:4630 00 00 00 00 00 00 00 00
.:4638 00 00 00 00 00 00 00 00
.:4640 00 00 00 00 00 00 02 A0
.:4648 00 0A C8 00 23 FA 00 2D
.:4650 5C 80 2D 5E 80 BD 1F 80
.:4658 2C 57 80 8D 7C 80 2F F3
.:4660 80 0B AA 00 02 80 00 00
.:4668 00 00 00 00 00 00 00 00
.:4670 00 00 00 00 00 00 00 00
.:4678 00 00 00 00 00 00 00 00
.:4680 02 AA 00 02 FE 80 0A F3
.:4688 80 2B 9E A0 2F 5D 80 29
.:4690 14 80 BD 57 E0 B5 45 E0
.:4698 8D 55 38 2F 17 E0 23 D3
.:46A0 80 2A DE 00 0A DF 80 DB
.:46A8 FE 00 0A E8 00 02 A0 00
.:46B0 00 80 00 00 00 00 00 00
.:46B8 00 00 00 00 00 00 00 00
.:46C0 02 A8 00 02 2A 80 0A AA
.:46C8 A0 2A AA A0 AA FA 80 2A
.:46D0 FE A0 A3 FF A0 2B D2 A0
.:46D8 2A DE A0 2A F6 A0 0A 9F
.:46E0 A0 2B 3E 80 2B CF 00 22
.:46E8 AB A0 2A AA A0 0A AB A0
.:46F0 0A 2A A0 00 8A A0 00 AA
.:46F8 80 00 2A 80 00 00 00 00
.:4700 18 00 00 A5 00 00 28 00
.:4708 00 00 00 00 00 00 00 00
.:4710 00 00 00 00 00 00 00 00
.:4718 00 00 00 00 00 00 00 00
.:4720 00 00 00 00 00 00 00 00
.:4728 00 00 00 00 00 00 00 00
.:4730 00 00 00 06 00 00 00 00
.:4738 00 00 00 00 00 00 00 00
.:4740 28 00 00 6A 00 00 18 00
.:4748 00 00 00 00 00 00 00 00
.:4750 00 00 00 00 00 00 00 00
.:4758 00 00 00 00 00 00 00 00
.:4760 00 00 00 00 00 00 00 00
.:4768 00 00 00 00 00 00 00 00
.:4770 00 00 00 00 00 00 00 00
.:4778 00 00 00 00 00 00 00 00
.:4780 27 00 00 5A 00 00 28 00
.:4788 00 00 00 00 00 00 00 00
.:4790 00 00 00 00 00 00 00 00
.:4798 00 00 00 00 00 00 00 00
.:47A0 00 00 00 00 00 00 00 00
.:47A8 00 00 00 00 00 00 00 00
.:47B0 00 00 00 00 00 00 00 00
.:47B8 00 00 00 00 00 00 00 00
.:47C0 28 00 00 A5 00 00 27 00
.:47C8 00 00 00 00 00 00 00 00
.:47D0 00 00 00 00 00 00 00 00
.:47D8 00 00 00 00 00 00 00 00
.:47E0 00 00 00 00 00 00 00 00
.:47E8 00 00 00 00 00 00 00 00
.:47F0 00 00 00 00 00 00 00 00
.:47F8 00 00 00 00 00 00 00 00
.:4800 00 00 01 00 00 03 00 00
.:4808 07 00 00 0F 00 00 1F 00
.:4810 00 1F 00 00 7F 00 03 7E
.:4818 00 07 FD 00 6F BB 00 FF
.:4820 B7 01 FF EF 03 FF F7 03
.:4828 FF FB 07 FF 0D 0F FF FE
.:4830 1F FF FF 3F FF FF 7F FF
.:4838 FF FF FF FF FF FF FF 00
.:4840 80 00 00 C0 00 00 E8 80
.:4848 00 FD C0 00 DF E0 00 BF
.:4850 F1 00 7F FB 80 FF FF C0
.:4858 FF FF E0 FF FF F0 FF FF
.:4860 F8 FF FF F8 FF FF FC FF
.:4868 FF FE FF FF FF FF FF FF
.:4870 7F FF FF BF FF FF DF FF
.:4878 FF EF FF FF F7 FF FF 00
.:4880 00 00 00 00 00 00 00 00
.:4888 00 00 00 00 00 00 01 00
.:4890 00 23 00 00 77 00 00 FF
.:4898 00 01 FB 00 03 FD 00 07
.:48A0 FE 00 4F FF 00 FF FF 41
.:48A8 FF FF E3 FF FF F7 FF FF
.:48B0 FF FF FF EF FF FF DF FF
.:48B8 FF BF FF FF 7F FF FF 00
.:48C0 10 00 00 38 00 00 7C 00
.:48C8 00 FE 00 00 FF 00 00 FF
.:48D0 80 00 FF C0 00 FF E0 00
.:48D8 FF F0 00 FF F8 00 FF F8
.:48E0 00 7F FC 00 BF FE 00 DF
.:48E8 FF 00 EF FF 80 EF FF C0
.:48F0 F7 FF E0 FB FF F8 FB FF
.:48F8 FE FD FF FF FE FF FF 00
.:4900 18 00 00 08 00 00 08 00
.:4908 00 08 00 00 0E 00 00 0E
.:4910 00 00 0E 00 00 FF F0 00
.:4918 F3 E0 00 75 C0 00 73 00
.:4920 00 3E 00 00 00 00 00 00
.:4928 00 00 00 00 00 00 00 00
.:4930 00 00 00 00 00 00 00 00
.:4938 00 00 00 00 00 00 00 00
.:4940 00 10 00 00 78 00 00 F8
.:4948 00 03 FC 00 07 FE 00 07
.:4950 FF 00 0F FF 00 0F FF 80
.:4958 1F FF C0 1F FF C0 3F FF
.:4960 E0 3F FF E0 3F FF E0 7F
.:4968 FF E0 7F FF E0 FF FF F0
```

```
.:4970 FF FF F8 FF FF F8 FF FF
.:4978 F8 FF FF FC FF FF FC 00
.:4980 04 00 00 0E 00 00 0F 00
.:4988 00 1F 80 00 3F 80 00 37
.:4990 C0 00 6F E0 00 EB 70 00
.:4998 F7 B0 00 EF F0 00 FF F0
.:49A0 00 00 00 00 00 00 00 00
.:49A8 00 00 00 00 00 00 00 00
.:49B0 00 00 00 00 00 00 00 00
.:49B8 00 00 00 00 00 00 00 00
.:49C0 18 00 00 08 00 00 18 00
.:49C8 00 3C 00 00 C2 00 00 C2
.:49D0 00 00 3C 00 00 18 00 00
.:49D8 08 00 00 18 00 00 00 00
.:49E0 00 00 00 00 00 00 00 00
.:49E8 00 00 00 00 00 00 00 00
.:49F0 00 00 00 00 00 00 00 00
.:49F8 00 00 00 00 00 00 00 00
.:4A00 0C 00 00 08 00 00 18 00
.:4A08 00 3C 00 00 C2 00 00 C2
.:4A10 00 00 3C 00 00 18 00 00
.:4A18 A8 00 00 58 00 00 00 00
.:4A20 00 00 00 00 00 00 00 00
.:4A28 00 00 00 00 00 00 00 00
.:4A30 00 00 00 00 00 00 00 00
.:4A38 00 00 00 00 00 00 00 00
.:4A40 59 00 00 49 00 00 2A 00
.:4A48 00 3E 00 00 2A 00 00 49
.:4A50 00 00 49 00 00 14 00 00
.:4A58 00 00 00 00 00 00 00 00
.:4A60 00 00 00 00 00 00 00 00
.:4A68 00 00 00 00 00 00 00 00
.:4A70 00 00 00 00 00 00 00 00
.:4A78 00 00 00 00 00 00 00 00
.:4A80 0C 00 00 6B 00 00 2A 00
.:4A88 00 7F 00 00 49 00 00 2A
.:4A90 00 00 6B 00 00 14 00 00
.:4A98 00 00 00 00 00 00 00 00
.:4AA0 00 00 00 00 00 00 00 00
.:4AA8 00 00 00 00 00 00 00 00
.:4AB0 00 00 00 00 00 00 00 00
.:4AB8 00 00 00 00 00 00 00 00
.:4AC0 0C 00 00 2A 00 00 2A 00
.:4AC8 00 7F 00 00 2A 00 00 2A
.:4AD0 00 00 2A 00 00 14 00 00
.:4AD8 00 00 00 00 00 00 00 00
.:4AE0 00 00 00 00 00 00 00 00
.:4AE8 00 00 00 00 00 00 00 00
.:4AF0 00 00 00 00 00 00 00 00
.:4AF8 00 00 00 00 00 00 00 00
.:4B00 18 00 00 08 00 00 2A 00
.:4B08 00 3E 00 00 2A 00 08 00
.:4B10 00 08 00 00 14 00 00 00
.:4B18 00 00 00 00 00 00 00 00
.:4B20 00 00 00 00 00 00 00 00
.:4B28 00 00 00 00 00 00 00 00
.:4B30 00 00 00 00 00 00 00 00
.:4B38 00 00 00 00 00 00 00 00
.:4B40 0C 00 00 6B 00 00 2A 00
.:4B48 00 7F 00 00 49 00 00 2A
.:4B50 00 00 6B 00 00 14 00 00
.:4B58 08 00 00 3C 00 00 00 00
.:4B60 00 00 00 00 00 00 00 00
.:4B68 00 00 00 00 00 00 00 00
.:4B70 00 00 00 00 00 00 00 00
.:4B78 00 00 00 00 00 00 00 00
.:4B80 0C 00 00 6B 00 00 2A 00
.:4B88 00 7F 00 00 49 00 00 2A
.:4B90 00 00 6B 00 00 14 00 00
.:4B98 08 00 00 3C 00 00 00 00
.:4BA0 00 00 00 00 00 00 00 00
.:4BA8 00 00 00 00 00 00 00 00
.:4BB0 00 00 00 00 00 00 00 00
.:4BB8 00 00 00 00 00 00 00 00
.:4BC0 01 00 00 03 80 00 1F C0
.:4BC8 00 3F E0 00 FF F0 00 FF
.:4BD0 E0 00 7F E0 00 7F C0 00
.:4BD8 3F 80 00 07 80 00 03 00
.:4BE0 00 00 00 00 00 00 00 00
.:4BE8 00 00 00 00 00 00 00 00
.:4BF0 00 00 00 00 00 00 00 00
.:4BF8 00 00 00 00 00 00 00 00
.:4C00 FF FF 00 00 FF FF 00 00
.
.
```

Listing C-31: The OOPLOT Subroutine Source Code

```
1000 ;PUT"@0:OOPLOT
1010 ;Y IN YPNT
1020 ;X/2 IN XPNT
1030 ;COLOR IN COLOR
1040 ;GRAPHICS BASE ADDRESS IN GBASE
1050 ;
1060 ;OR'S A POINT
1070 ;
1080 ;
1090 ;
1100 ;
1110 ;
```

```
;
OPLOT   ANOP
        LDA YPNT
        CMP #$C7
        BCS RTR1
        TAX
        LDA XPNT
        CMP #$A0
        BCS RTR1
        LSR A
        LSR A
        TAY
        LDA GBASE
        CLC
        ADC VLKUPL,X
        STA POINT
        LDA GBASE+1
        ADC VLKUPH,X
        STA POINT+1
        LDA POINT
        CLC
        ADC HLKUPL,Y
        STA POINT
        LDA POINT+1
        ADC HLKUPH,Y
        STA POINT+1
;
        LDA XPNT
        AND #$03          ;STRIP BIT POS
        TAX
        LDY #$00
        LDA (POINT),Y
        AND PAND,X        ;CLEAR BIT
        STA CTEMP
        LDY COLOR
        LDA COLK,Y
        AND POR,X
        ORA CTEMP
        LDY #$00
        STA (POINT),Y
RTR1    RTS
;
POR     .BYTE $C0,$30,$0C,$03
PAND    .BYTE $3F,$CF,$F3,$FC
COLK    .BYTE $00,$55,$AA,$FF
;
        .END
```

Glossary

accumulator—An internal register in the microprocessor that is used by all arithmetic instructions and most data transfer instructions.

assembler—A program that translates mneumonics for machine language instructions into machine language.

bank—One of four 16K blocks of memory in the Commodore 64.

bit—One binary digit. A bit can only have the values 0 and 1.

bitmapped graphics—A graphics mode in which each pixel on the screen is represented by one or more bits in memory.

byte—A grouping of eight bits. A byte can represent any value from 0 to 255. The Commodore 64 operates on memory one byte at a time.

color RAM—A 1K area in memory, starting at $D800, in which each byte represents the color for one character on the screen.

coresident assembler—An assembler program that has its editor, assembler, loader, and monitor in memory at the same time.

disassembly—A recreation of assembly language source code created by reading the object code and assigning the mneumonics to the code. This is the reverse of assembling a program.

flicker fusion frequency—24 cycles per second is the frequency at which the human eye will merge the individual screens being shown into continuous movement.

graphics memory—An area in memory where the bitmapped graphics images are stored.

HEX—Abbreviation for hexadecimal.

HEX file—An intermediate file produced by the assembler that can be loaded into memory using a loader program or transferred to another machine.

hexadecimal—The base 16 numbering system. In this numbering system, a digit can have a value of 0 to 15. Numbers above 9 are represented by the letters A through F.

interrupt—A signal that causes the microprocessor to stop executing the current program and execute another program in memory. The address where the first program was stopped is stored on the stack as well as the status register, so that program execution can continue at a later time.

interrupt vector—Two bytes in memory that contain the address of the routine to be executed when an interrupt occurs.

KERNAL—A set of machine language subroutines in ROM that can be called by your program to perform complex tasks. None of the KERNAL routines in the Commodore 64 are useful for games.

label—A string of characters that can be used to represent memory locations or data.

loader—A program that translates a HEX file into binary data and stores it into memory.

LSB—The least significant bit in a byte.

macro—A user-definable macroinstruction made up of one or more instructions. It is treated like any other instruction during programming and expanded into its component instructions during assembly.

maskable interrupt—An interrupt that can be disabled through software.

mneumonics—Short character strings that are used as memory aids for assembly language instructions.

monitor—A program that is used to examine and change memory locations. It also can be used to load and save areas of memory.

MSB—The most significant bit in a byte.

multiplexing—The technique of using one sprite to display two different images. The different images may overlap.

nibble—A grouping of four bits. A nibble can be represented by one hexadecimal digit.

object code—Binary data that can be directly executed by the microprocessor.

opcode—A value that represents the instruction to be executed.

operand—The data that is to be operated on by the instruction.

pixel—The smallest dot that can be generated by the computer to be displayed on the screen.

program counter—An internal 16 bit register that is used to access the next memory location.

screen—One refresh of the video display. The display is refreshed 60 times per second.

scroll—Smooth movement of a portion of the screen.

source code—Text that represents an assembly language program. This will be translated into machine code by the assembler.

sprite—A small object that can be moved independently of the background.

stack—An area of memory that is used by the microprocessor for temporary storage.

stack pointer—A byte that is used as an index into the stack area.

status register—An internal register whose bits are affected by the different instructions. Bits in the status register can be tested by branching instructions.

touch pad—A device that plugs into the joystick port on the computer, and when touched on its surface, returns a X and Y coordinates representing the position on the pad where the touch occurred.

zero page—The area in RAM from $0000 to $00FF, which can be accessed by the microprocessor faster than any other area in memory. A program that uses zero page addressing extensively can be 2/3 the size that the program would be otherwise.

Index

Index

<, 4, 5
.BYTE, 4, 51
.DBYTE, 4
.END, 4
.LIB, 4, 20, 65
.MAC, 5
.MND, 5
.WORD, 4
6510 chip, 9
6526 chip, 11
6567 chip, 11
6591 chip, 11

A

absolute addresses, 7
absolute commands, 15
absolute indexed addressing, 15
accumulator, 11
ADC instruction, 78
address bus, 22
address lines, 11
addressing modes, 14
ADRES macro, 100
AND instruction, 78
animating sprites in BOGHOP, 73
animation, 2
ANOP macro, 100
Architecture, 6510, 11
arithmetic instructions, 13, 14
ASL instruction, 79
ASL2 macro, 100
assembler, 4
assembler, choosing an, 6
assembler, functions of, 5
assembly language, 4

B

bad guys in BOGHOP, 71
band pass filter, 39
BANK macro, 24, 101
bank switching, 11, 22
banks of memory, 23
BASE address register, 27
BASIC, 3
BASIC program, 120
BCC instruction, 79
BCOLO register, 29
BCS instruction, 80
BEQ instruction, 80
BGT macro, 101
BIN, 120
binary data, 4
binary data file, 120
binary files, 119
bit, 9
BIT instruction, 79
bit mapping, 29
bitmapped graphics grid, 48
blanking the screen, 33
BLE macro, 101
BLNK macro, 102
BMI instruction, 80
BNE instruction, 81
BOGDAT file, 65
BOGDAT source code, 243
BOGDEF, 65
BOGDEF file, 64
BOGDEF source code, 241
BOGHOP game, 64
BOGHOP program, 64
BOGHOP program source code, 207
BOGHOP.O program, 233
BOGSPR file, 249
BPL instruction, 81
BPRIOR, 32
BREAK instruction, 81
buffer areas, 67
bugs, 18
BVC instruction, 82
BVS instruction, 82
byte, 4, 9

C

cartridge monitors, 8
CENTIPEDE, 58
central processing unit, 11
chaining, 18
character generator ROM, 25
character generator section, 25
character graphics grid, 50
character sets, custom, 28
characters sets, custom, 28
chip, sound interface device, 11
chip, video interface, 11
chips, 10
CINV, 20
CLBACK1, 56
CLBACK1 file, 169
CLC instruction, 82
CLD instruction, 83
CLI instruction, 83
CLSP1, 56
CLSP1 file, 169
CLSP2, 55
CLSP2 file, 167
CLV instruction, 83
CMP instruction, 84
collision detection, 32
collision status register, 19
color on televisions, 51
color RAM, 27
color, sprite, 31
COM-KO program, 54, 153
commands, assembly language, 6
comments, 13
Commodore's Macro Assembler Development System, 4
COMMON file, 68
COMMON file, 41, 65
COMMON source code, 142
complex interface adapter chips, 11
concepts, game, 61
conditional branching instructions, 14
control lines, 22
controls, sprite, 31
coresident assembler, 6
CPX instruction, 84
CPY instruction, 85

D

data areas, defining, 6
data definitions, 18
DATA file, 41
data movement, 13
data movement instructions, 14
data section of BOGHOP, 68
DATA source code, 144
data, tables of, 6
DBADC macro, 102
DBADC1 macro, 103
DBDEC macro, 5
DBINC, 103
DBINC macro, 103
DBPL macro, 103
DBSBC macro, 104
DBSBCI macro, 104
DDRA and DDRB registers, 34
debugging, 18
DEC instruction, 85
defining the system, 70
definitions, RAM, 66
demonstration, sound generator, 41
designing video games, 57
DEX instruction, 86
DEY instruction, 86
difficulty levels, 62
disassembly, 5
DISPLAY PIC program, 55, 153
display, video, 1
DONKEY-KONG, 58
DS macro, 104
DS macro, 65

E

EDIT SND program source code, 145
editor, sound, 44
editor, text, 6
electron beam, 1
elements of game design, 61
enabling a sprite, 31
EOR instruction, 86
equates statements, 69
expansion, data, 7
expansion, macro, 7
expansion, sprite, 31

F

FILBYT macro, 105
FILL macro, 105
filtering sounds, 36
flicker fusion frequency, 14
flow of control instructions, 2, 13
frame, 2
fusion frequency, 2

G

glossary, 254
GRABAS macro, 27, 105
GRAPH macro, 106
graphics tablet, 52
graphics, hand coding, 47
grids, graphic, 50

H

hardware, 10
hardware registers, 26
harmonic content, 39
HDEC macro, 106
hex file, 7
hexadecimal, 10
high pass filter, 39
HINC macro, 106

I

immediate mode addressing, 14
implied addressing, 16
INC instruction, 87
indirect addressing, 16
indirect addressing with indexes, 15
initializing RAM, 70
instruction set 6510, 77
instruction types, 13
INT DEMO.O program, 133
INT1, 20
INTER-DEMO source code, 132
interpreter, BASIC, 3
interrupt demonstration, 20
interrupt routines in BOGHOP, 74
interrupts, 19
interrupts, video, 33
INX instruction, 87
INY instruction, 87
IPULL macro, 107

J

JCC macro, 107
JCS macro, 107
JEQ macro, 108
JGE macro, 108
JGT macro, 108
JLE macro, 109
JLT macro, 109
JMI macro, 109
JMP instruction, 88
JNE macro, 110
JOY1 and JOY2 registers, 34
joystick, 55
joysticks, 34, 74
JPL macro, 110
JSR, 6
JSR instruction, 88

K

KERNAL routines, 22
KILL macro, 20, 112
KO-COM program, 54, 153
Koala Pad, 54

L

labels, 13
LDA instruction, 88
LDMEM macro, 110
LDMEMX macro, 111
LDMEMY macro, 111
LDX instruction, 89
LDY instruction, 89
least significant bit, 9
level of play, BOGHOP, 71
LEVEL register, 66
levels, difficulty, 62
loader, 7
LOOKUP data file, 248
LOOKUP file, 65
low pass filter, 39
LSR instruction, 90
LSR2 macro, 111

M

machine language, 3
MACLIB file, 64, 99
MACLIB source code, 121
Macro Assembler Development System, 4
macro library, 17, 65
macro-instructions, 5
macros, 5, 21, 99
main program, 1,718
main program, BOGHOP, 70
mask register, video interrupt, 33
maskable interrupt, 20
memory banks, 23
memory dumps, 119
memory locations, 4
memory maps, selecting, 22
microprocessor, 9
MLTSP register, 31
mnemonics, 4
mode, bitmapped, 29
modes, addressing, 14
modules, separate, 18
monitor, 1
monitor, assembly language, 7
most significant bit, 9
movement subroutines in BOGHOP, 76
multicolor bitmapped mode, 30
multicolor graphics grids, 49
multicolor mode, 28
multiplexing, 2
multipliers, sprite, 31
MULTOF macro, 30, 112
MULTON macro, 30, 112
MVCOL macro, 27, 113
MVIT subroutine, 154
MVMEM macro, 113

N

names, assignment of, 5
nibble, 9
NIBLL macro, 113
NIBLR macro, 114
NMINV, 20

noise, 36
nonmaskable interrupts, 20
NOP instruction, 76, 90
NOT macro, 114
NOTPT register, 66
NOTTM register, 66

O

OOPLOT file, 65
OOPLOT subroutine source code, 252
opcode, 13
opcodes, 4
operand, 13
operating system, disabling, 20
OPTION register, 66
ORA instruction, 91

P

PAC-MAN, 57
packages, graphic, 5,253
PEEK, 4
PHA instruction, 91
PHOENIX V1.4N program, 169
phosphors, 1
PHP instruction, 91
PIC A CASTLE, 55
PIC A CASTLE Koala Pad picture file, 154
pixel, 1
PLA instruction, 92
players in BOGHOP, 72
PLP instruction, 92
point plotting routine, 69
pointer, stack, 12
pointers, sprite, 30
POKE, 4
positioning sprites, 31
priorities, sprite, 32
priority, sprite, 30
program counter, 11
pseudo opcodes, 4
pulse, sync, 1
PUNPCK macro, 114
purposes of games, 61

Q

QDDEC macro, 115
QDINC macro, 115

R

RAM, 11
RAM definition, 66
RAM definitions, 17
RAM registers, 7
RAND registers, 66
RANSEC, 66
RAST macro, 20, 115
raster interrupt, 19, 20
RASTER register, 33
register names, 21
registers, X and Y, 11
relative addressing, 16
relocation of loading address, 7
resetting, 20
Revenge of the Phoenix, 59
Revenge of the Phoenix game program, 169
ROL instruction, 92
ROM, 11
ROR instruction, 93
routines, 18
RTI instruction, 93
RTS instruction, 93

S

SBC instruction, 94
scores, 70
scoring, 63
screen, 2
screen maker utility, 56
SCREEN timer, 66
SCREEN-MAKE program, 167
scrolling, 33
SEC instruction, 94
SED instruction, 95
SEI instruction, 95
setting sprite color, 31
shots in BOGHOP, 72
SLIB.O, 55, 56
SLIB.O file, 166
SMNPC macro, 116
SMNPCX macro, 116
SNDDEF file, 41
SNDDEF source code, 143
SNDEF file, 41
SNDTM registers, 66
software, 53
SOUND DEMO program, 139
SOUND EDIT program, 151
sound editor, 41
sound effects in games, 62
sound interface device chip, 11
SOUND program source code, 133
source code file, 120
source code files, 119
speed of assembly, 6, 7
sprite grid, unexpanded, 51
sprite grids, expanded, 52
Sprite Maker, 55
SPRITE MAKER PROGRAM, 165
sprites, 30
SPRXSZ and SPRYSZ registers, 51
SRC, 120
SSCOL, 33
SSCOL register, 32
STA instruction, 95
stack, 12
standard text mode, 24
status register, 12
STX instruction, 96
STY instruction, 96
subroutines, 6, 18
symbol table, 7
sync pulse, 1
SYSDEF, 21
SYSDEF file, 64
SYSDEF source code, 131
system definitions, 17

T

TAX instruction, 96
TAY instruction, 97
televisions, 1, 51
testing instructions, 13, 14
TEXT macro, 116
text editor, 6
text memory, 25
TEXT mode, 27
tones, 36
TPDEC macro, 117
TPINC macro, 117
TSX instruction, 97
TXA instruction, 98
TXBAS macro, 27, 17
TXS instruction, 97
TYA instruction, 98

U

UNBLNK macro, 118
UNPACK macro, 118
update time, 2

V

vertical sync pulse, 1
VIC chip, 23
VIC II, 11
VIDBAS, 27
VIDBAS register, 24, 25, 27
video display, 1
video games, designing, 57
video interface chip, 11
video interrupts, 33
VIRQ register, 33
VIRQM register, 33
visual impact of games, 62

W

waveforms, 36

X

XSCRL register, 33
XXPLOT, 69
XXPLOT file, 65
XXPLOT subroutine source code, 247

Y

YSCRL register, 29, 33

Z

zero page addressing, 14
zero page indexed addressing, 15

Edited by Marilyn L. Johnson

Commodore 64™ Assembly Language Arcade Game Programming

by Steve Bress

Discover the secrets of writing your own pro-quality, fast-action, animated arcade games!

Unlock an exciting new dimension of your C-64's graphics capabilities that's just not possible when you program in BASIC! Discover how you can create amazingly sophisticated arcade-type games featuring the fast-paced, animated action and sophisticated sounds used in arcade video machines and commercially produced computer games retailing for upwards of $40! Even find out how you can use assembly language game techniques to better control your C-64 in all types of programming applications!

Everything you need is here in this learn-by-doing handbook. It gives you a firm understanding of how your C-64 functions—the VIC II and SID chips, the way memory is banked, how graphics are controlled, collision detection, interrupts, and more. You'll find numerous techniques that will help you easily solve initial game designing problems plus dozens of subroutines, utilities, tips, and tricks that can save you hours of coding time when you are creating games or almost any other kind of assembly language-based program.

There's even a complete set of macros that makes the design of sprites and animation far easier and more productive than anything you've ever been able to accomplish using BASIC. Creation of music and sound effects is also thoroughly explained. And, using the two full-scale example programs (Revenge of the Phoenix and Boghop) as your guide, you'll be designing your own animated arcade games in record time.

If you've ever wanted to try your hand at writing fast-paced arcade action games for your C-64, you'll find this a truly exciting sourcebook. It shows you how to effectively come up with game ideas, how to plan your game scenario, how to code, and finally to test your game to make sure it's working the way you want it to!

Steve Bress is the co-owner of a company that specializes in custom interface boards for home computers. He has extensive programming experience for home computers ranging from the ATARI 2600 to the IBM PC!

TAB BOOKS Inc.

Blue Ridge Summit, Pa. 17214

FPT >$ 14.95 ISBN 0-8306-1919-4

PRICES HIGHER IN CANADA 1445-0585